

Wavelet toolware

Software
for
Wavelet
Training

Charles R. Chaf,
Andrew R. Chen,
& C. Wayne Liu



Wavelet Toolware: Software for Wavelet Training

This page intentionally left blank

Wavelet Toolware: Software for Wavelet Training

Andrew K. Chan

Department of Electrical Engineering
Texas A&M University

Steve J. Liu

Department of Computer Science
Texas A&M University



San Diego London Boston
New York Sydney Tokyo Toronto

This book is printed on acid free paper. (∞)

Copyright © 1998 by Academic Press

All rights reserved.

No part of this publication may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopy, recording, or any information storage and retrieval system, without permission in writing from the publisher.

ACADEMIC PRESS

525 B Street, Suite 1900, San Diego, CA 92101-4495, USA

1300 Boylston Street, Chestnut Hill, MA 02167, USA

<http://www.apnet.com>

ACADEMIC PRESS LIMITED

24–28 Oval Road, London NW1 7DX, UK

<http://www.hbuk.co.uk/ap/>

Printed in the United States of America

98 99 00 01 02 IP 9 8 7 6 5 4 3 2 1

Disclaimer:

This eBook does not include the ancillary media that was packaged with the original printed version of the book.

Contents

Part I	ii
1.1 From Fourier Analysis to Wavelet Analysis	1
1.2 Short-Time Fourier Transform	2
1.3 Window Measures	3
1.4 What Is a Wavelet?	5
1.4.1 Wavelets at different resolutions (scales)	6
1.5 Continuous Wavelet Transform	7
1.6 Multiresolution Analysis	9
1.6.1 Two-scale relations	12
1.6.2 Decomposition relation	13
1.6.3 Development of the decomposition and reconstruction algorithms	15
1.6.4 Mapping of functions into the approximation space	16
1.7 Types of Wavelets	18
1.7.1 Orthonormal wavelet bases	18
1.7.2 Semi-orthogonal wavelet bases	19
1.7.3 Biorthogonal wavelet bases	21
1.8 Wavelet Packets	22
1.8.1 Wavelet packet algorithms	23
1.9 Two-Dimensional Wavelets	24
1.9.1 Two-dimensional wavelet packets	26
 Part II	 27
2.1 Wavelet Algorithms Overview	29
2.2 Algorithm for Computing the STFT	29
2.3 Algorithm for Computing the CWT	31
2.4 Computation and Display of Scaling Functions	32

2.5	Computation and Graphical Display of Wavelets	34
2.6	Mapping between B-Splines and Their Duals	35
2.7	Decomposition of a 1-D Signal	36
2.8	Reconstruction of 1-D Signals	37
2.9	Thresholding	37
2.9.1	Hard thresholding	38
2.9.2	Soft thresholding	38
2.9.3	Quantile thresholding	38
2.9.4	Universal thresholding	38
2.10	Two-Dimensional Extension of Wavelet Algorithms . . .	39
2.11	Other Algorithms	40

Part III		41
3.1	The Wavelet Toolware Overview	43
3.1.1	Installation of Toolware	43
3.1.2	Wavelets	44
3.2	Using Toolware	45
3.3	2-D DWT Tool	46
3.3.1	Data file selection	47
3.3.2	Wavelet decomposition	49
3.3.3	Display thresholding	50
3.3.4	Wavelet reconstruction	51
3.4	1-D DWT Tool	55
3.4.1	1-D DWT decomposition	56
3.4.2	Viewing and processing the data	57
3.5	Continuous Wavelet Transform Tool	62
3.6	Short Time Fourier Transform Tool	63
3.7	Wavelet Builder Tool	64

Preface

Wavelet Toolware is a companion software package to the book *An Introduction to Wavelets* by Charles K. Chui. It is designed for the reader to gain some hands-on practice in the subject of wavelets. The objective is to provide basic signal analysis and synthesis tools that are flexible enough for the reader to easily increase the capability of the Toolware by adding processing algorithms to tailor to specific areas of application. Wavelet Toolware contains, among other algorithms and codes, the one-dimensional (1-D) and two-dimensional (2-D) Discrete Wavelet Transforms (DWTs) and their inverse transforms, as well as computations of the Continuous Wavelet Transform (CWT) and the Short Time Fourier Transform (STFT). Simple signal processing applications are also included for reader to practice with the software.

This manual for Wavelet Toolware leads the reader through the basics of wavelet theory and shows how it can be applied to a number of engineering problems. It is organized into three major divisions. Part I concerns the basics of wavelet theory, and readers should get a very fundamental understanding of the development of the theory and gain appreciation of the time-frequency (or time-scale) representation of signals. Part II contains several major algorithms that correspond to the topics discussed in Part I. In most cases, the algorithms are outlined in procedural steps. Finally, Part III provides hands-on practice with the Wavelet Toolware. It contains many practice sections ranging from installation of the software to denoising of images.

This manual is designed to be self-contained for the reader who is unfamiliar with wavelet theory but wants to gain a basic understanding and the usage of wavelet theory. After thoroughly studying this manual and practicing with the Toolware, readers should be able to apply the basic aspects of wavelet analysis to problems of their respective disciplines.

Andrew K. Chan
Steve J. Liu

College Station, Texas
February 13, 1998

This page intentionally left blank

Acknowledgments

The developmental work of this software for training in wavelet algorithms is based on our experience in teaching short courses for the Texas Engineering Extension Services over the last few years. Our gratitude goes to Professor Charles K. Chui for his organization of the short courses, help in the review of this manuscript, and many valuable suggestions. Some of the computer code evaluations and graphical illustrations were carried out by our former and current students, Howard Choe, Hung-Ju Lee, Nai-wen Lin, Dongkyoo Shin, Hsien-Hsun Wu, Michael Yu, Xiqiang Zhi. We thank them for their careful and detailed work. Special thanks are due to Brandon Scott Wadsworth, who spent countless hours perfecting the GUI of the software. We are indebted to Margaret Chui for her help in the editorial work of this manuscript.

This page intentionally left blank

To
Jesus Christ our Lord, and
To
our wives, Sophia and Grace.

This page intentionally left blank

Part I

Wavelet Theory

This page intentionally left blank

1.1 From Fourier Analysis to Wavelet Analysis

Before the work of Haar [1], a continuous function (or an analog signal) $f(t)$ was represented either by an entire series (polynomial basis) or by a Fourier series (sinusoidal basis). Since both bases functions have infinite time duration ($-\infty < t < \infty$), it is awkward to use any of these bases to represent finite time-domain (transient) signals. Haar's work demonstrated that a continuous function can be approximated arbitrarily closely by an orthonormal basis with local support (the Haar basis). Since then, mathematicians have constructed various new bases to represent a variety of analog functions.

During the past three decades, signal processing has grown to become a major discipline in engineering and computing science. Since most finite energy signals, either natural or otherwise, are transient (or nonstationary) in nature, it is most natural and effective to represent these signals by localized finite-energy bases. Because wavelets belong to this class of bases, the development of wavelet theory and its applications to signal processing in various engineering disciplines has gained tremendous popularity in recent years.

Fourier analysis has been the major mathematical tool for signal representation and processing for at least the past 50 years. The discrete version of this approach, called Fourier series, breaks down a given signal into sinusoidal functions with different harmonics of the fundamental frequency. Since sinusoidal signals are periodic signals, Fourier analysis is an excellent tool for analyzing this class of signals. However, it is inefficient for representing transient signals. Recall that the definition of the Fourier transform of an analog function $f(t)$ is defined by

$$\hat{f}(\omega) = \int_{-\infty}^{\infty} f(t)e^{-j\omega t} dt. \quad (1.1)$$

It is easy to show that if $f(t)$ is replaced by a delta function $\delta(t - t_0)$ at $t = t_0$, then the Fourier transform (or spectrum) becomes a sinusoidal function $e^{-j\omega t_0} = \cos t_0\omega - j \sin t_0\omega$. Hence, it takes an infinite number of frequencies to represent a signal that exists at only one point in the time domain. On the other hand, if the function to be analyzed is a sinusoidal function of a single frequency, the spectrum is a delta function. A careful study of this consideration leads to discrete Fourier analysis. Discrete Fourier analysis is very efficient for studying global periodic function. However, for transient signals, such as music, image, speech, acoustic noise, seismic signals, thunder, and lightning, it is more efficient to represent these signals by localized finite energy bases.

Other drawbacks of the Fourier analysis include the following:

1. The Fourier transform can be computed for only one frequency at a time.
2. Exact representations cannot be computed in real time.
3. The Fourier transform provides information only in the frequency domain, but none in the time domain.

1.2 Short-Time Fourier Transform

Fourier analysis, being unable to provide localized time and frequency information simultaneously, becomes the most serious drawback for processing of transient signals. In order to gain information from both time and frequency domains simultaneously, engineers have used the Short-Time Fourier Transform (STFT) to accomplish their goal, even if only approximately.

The STFT is formally defined by the integral transform

$$ST_w f(t, \omega) = \int_{-\infty}^{\infty} f(\tau) \overline{w(\tau - t)} e^{-j\omega\tau} d\tau, \quad (1.2)$$

where the overbar indicates complex conjugation. The function $w(t)$ is called the window function to be chosen by the user. This is the reason that the STFT is also sometimes called the Windowed Fourier Transform (WFT). The variables (t, ω) are the transform variables from the single time domain variable τ . Hence, the STFT transforms a one-variable function into a two-variable function. It provides time-frequency information of the function $f(t)$ on the time-frequency $(t-f)$ plane. The complex spectrum of $ST_w f(t, \omega)$ gives approximated spectral properties of the function near the time location t . It is only an approximation, since the expression in (1.2) is the Fourier transform of the product of the functions $f(\tau)$ and $w(\tau - t)$. In other words, (1.2) gives the spectrum of the product function $f(\tau)w(\tau - t)$ instead of the function $f(\tau)$ alone. Even if the window function $w(t)$ is a rectangular window with unit amplitude and $f(\tau)$ is a pure sinusoidal function $\cos(\omega_0 t)$, the transform of the product $f(\tau)w(\tau - t)$ is not just a delta function in the spectrum. Rather it is the convolution of the delta spectrum with the sinc function spectrum of the rectangular function. This broadening of the delta function $\delta(\omega - \omega_0)$ to a $\text{sinc}(\omega - \omega_0)$ function represents the effect of the truncation due to the window function. We call that a windowing effect on the function $f(\tau)$. We will see in

later sections that the accuracy of the information on the t - f plane is controlled by the widths of the window in time and frequency domains. In addition, the uncertainty principle governing the area of the window in the t - f plane places a limit on the resolution that one can achieve using STFT.

The spectral domain equivalent of (1.2) comes from the Parseval identity, which gives

$$\begin{aligned} ST_w f(t, \omega) &= \int_{-\infty}^{\infty} f(\tau) \overline{w(\tau - t)} e^{-j\omega\tau} d\tau. \\ &= \frac{1}{2\pi} e^{-j\omega t} \int_{-\infty}^{\infty} \hat{f}(\theta) \overline{\hat{w}(\theta - \omega)} e^{j\theta t} d\theta. \end{aligned} \quad (1.3)$$

Expression (1.3) indicates that the time-domain windowing process is also a spectral-domain windowing process. It provides information on the spectral energy of the signal near the center of the window.

The original signal $f(t)$ can be recovered from the STFT uniquely. Hence the STFT is a unique transformation. The recovery formula for the STFT is given by

$$f(t) = \frac{1}{2\pi \|w\|} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} ST_w(\tau, \omega) w(\tau - t) e^{i\omega\tau} d\omega d\tau, \quad (1.4)$$

where $\|w\|$ is the norm of the function that will be defined in the next section.

1.3 Window Measures

In order to make an intelligent choice of a window function for the STFT, some measures of “goodness” must be established to evaluate the suitability of the window function. Engineering measurements such as the main lobe zero-crossing and the 3db points do not gauge the concentration of energy of the function. In the wavelet literature, the RMS width of the function is used to measure energy concentration in the time and frequency domains. These measurements are explicitly expressed respectively as

$$\Delta w = \frac{1}{\|w\|} \left\{ \int_{-\infty}^{\infty} (t - t^*)^2 |w(t)|^2 dt \right\}^{\frac{1}{2}} \quad (1.5)$$

$$t^* = \frac{1}{\|w\|_+^2} \left\{ \int_{-\infty}^{\infty} t |w(t)|^2 dt \right\} \quad (1.6)$$

and

$$\Delta \hat{w}^+ = \frac{1}{\|\hat{w}\|} \left\{ \int_0^\infty (\omega - \omega^{*+})^2 |\hat{w}(\omega)|^2 d\omega \right\}^{\frac{1}{2}} \quad (1.7)$$

$$\omega^{*+} = \frac{1}{\|\hat{w}\|_+^2} \left\{ \int_0^\infty \omega |\hat{w}(\omega)|^2 d\omega \right\}. \quad (1.8)$$

Here, the usual definition of the energy norms

$$\|w\| = \left\{ \int_{-\infty}^\infty |w(t)|^2 dt \right\}^{\frac{1}{2}} \quad (1.9)$$

$$\|\hat{w}\|_+ = \left\{ \int_0^\infty |\hat{w}(\omega)|^2 d\omega \right\}^{\frac{1}{2}} \quad (1.10)$$

applies to these formulas. The measurement Δw is considered as half of the time-domain window width, which measures the concentration of energy in the window. A smaller window width means more energy is concentrated in a smaller area. One may use a small time window width to deduce the fine structure of a signal. On the other hand, $\Delta \hat{w}^+$ is the measurement of energy concentration in the spectral domain. In addition, t^* and ω^{*+} are the centers of gravity of the window in the time and spectral domains, respectively. If $w(t)$ is real and even, and the average value of the function is zero, then $\hat{w}(\omega)$ is also symmetric with respect to the origin and $\hat{w}(0) = 0$. In this case, the integration in the frequency domain expression extends only from 0 to ∞ , and by the definition of (1.8), ω^* is located on the ω -axis other than the origin. In general, one must compute the window widths numerically, since most of the window functions we are considering in this manual may not be described analytically.

We can easily see that the window area of the STFT in the t - f plane is given by

$$[(t + t^* - \Delta w), (t + t^* + \Delta w)] \times [(\omega + \omega^* - \Delta \hat{w}), (\omega + \omega^* + \Delta \hat{w})]. \quad (1.11)$$

The center of the window is located at $(t + t^*, \omega + \omega^*)$ on the t - f plane where t is the time parameter being considered. Once we have chosen the window function $w(t)$, the window widths $2\Delta w$ and $2\Delta \hat{w}$ are fixed everywhere in the t - f plane since they are independent of t and ω . This effect is demonstrated on the t - f plane as given in Fig 1.1. This characteristic presents a distinct disadvantage of the STFT since it creates difficulty in accurately processing signals with a time-dependent spectrum, such as a chirp signal. In order to achieve a high degree of accuracy, the STFT must be repeatedly applied to the signal with a varying window width each time.

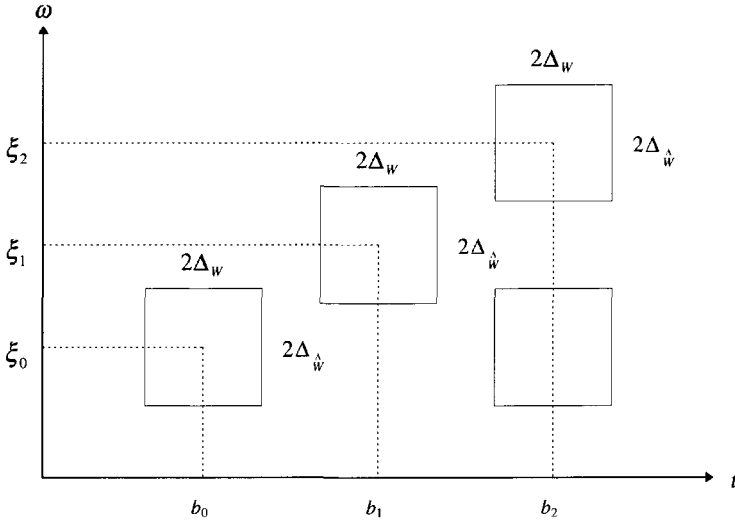


Figure 1.1: STFT windows

1.4 What Is a Wavelet?

Wavelets are finite-energy functions with localization properties that can be used very efficiently to represent transient signals. Efficiency here means only a small finite number of coefficients are needed to represent a complicated signal. In contrast with the sinusoidal functions of infinite extent (big waves), “wavelet” implies a “a small wave.” Strictly speaking, it also means mathematically that the area under the graph of the wavelet $\psi(t)$ is zero, i.e., $\int_{-\infty}^{\infty} \psi(t)dt = 0$. In the spectral domain, this property is equivalent to $\hat{\psi}(0) = 0$, i.e., the spectrum of the wavelet has a value of zero at $\omega = 0$. In engineering terms, the wavelet has no d-c offset. This spectral-domain behavior makes the wavelet a bandpass filter. The reader can use the Toolware to generate the graph of any wavelet listed in the Part III by following the instructions given later in this manual.

It is easy to see from any wavelet and its spectrum that the energy of the wavelet is concentrated in a certain region of both the t - and ω -axes. This localization property is an important feature of the wavelets. If a wavelet is more localized (that is, the energy of the wavelet is concentrated in a smaller region), it produces a better representation of the signal in the time-frequency (or time-scale) plane. In our case, “better” means higher resolution and requires less coefficients in the

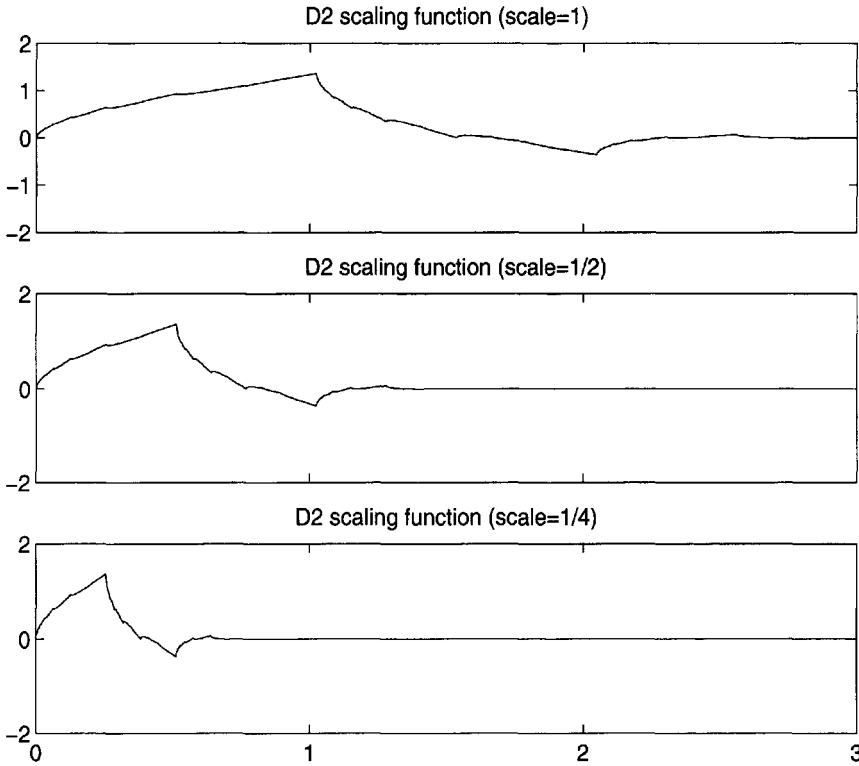


Figure 1.2: The scaling functions of the Daubechies-2 wavelet at three different scales

representation.

1.4.1 Wavelets at different resolutions (scales)

For a given wavelet $\psi(t)$, a scaled and translated version is designated by

$$\psi_{ba}(t) = \frac{1}{\sqrt{a}} \psi\left(\frac{t-b}{a}\right). \quad (1.12)$$

The parameter a corresponds to the scale while b is the translation parameter. The wavelet $\psi_{01}(t) = \psi(t)$ is called the basic wavelet (or mother wavelet). The Daubechies D_2 scaling function at three different scales is shown in Fig 1.2.

It is important to note that the shape of the wavelet remains the same under translation and scaling. We will show below that with the proper choice of (b, a) , the set of wavelets $\psi_{ba}(t)$ constitutes the basis functions of a Reisz (or stable) basis in the L^2 (or finite-energy) space. Keeping this concept of basis in mind, we will show that the idea of

wavelet signal processing is not much different from that of Fourier processing, at least when “orthonormal wavelet” are considered. Instead of decomposing a signal into sinusoidal functions of different frequencies (the Fourier basis), wavelet signal processing seeks to decompose a transient signal into a linear combination of a scaled and translated version of the basic wavelet (the wavelet basis).

1.5 Continuous Wavelet Transform

The continuous (integral) wavelet transform (CWT/IWT) of a signal $x(t)$ is a linear transform defined by the integral

$$\begin{aligned} W_\psi x(b, a) &= \int_{-\infty}^{\infty} x(t) \overline{\psi_{ba}(t)} dt \\ &= \frac{1}{\sqrt{a}} \int_{-\infty}^{\infty} x(t) \overline{\psi\left(\frac{t-b}{a}\right)} dt \\ &= \langle x(t), \psi_{ba}(t) \rangle. \end{aligned} \quad (1.13)$$

The last expression of (1.13) represents the inner product of two functions, defined in the L^2 space by

$$\langle f, g \rangle = \int_{-\infty}^{\infty} f(t) \overline{g(t)} dt. \quad (1.14)$$

Referring to (1.13), the CWT computes, via the inner product formula, the wavelet coefficient of $x(t)$ associated with the wavelet $\psi_{ba}(t)$. This coefficient indicates the correlation between the function $x(t)$ to the wavelet $\psi_{ba}(t)$. Higher correlation produces a larger coefficient. An example of the CWT is shown in Fig 1.3.

For the sake of comparison, we write the STFT in the inner product form as

$$\begin{aligned} ST_w(x) &= \int_{-\infty}^{\infty} x(t) \overline{w(t-b)} e^{-j\omega t} dt \\ &= \langle x(t), w(t-b) e^{j\omega t} \rangle. \end{aligned} \quad (1.15)$$

One immediately sees from (1.13) and (1.15) that the difference between CWT and STFT is that the kernels for these two transforms are different. CWT and STFT are interchangeable if we switch

$$w(t-b) e^{j\omega t} \leftrightarrow \psi_{ba}(t). \quad (1.16)$$

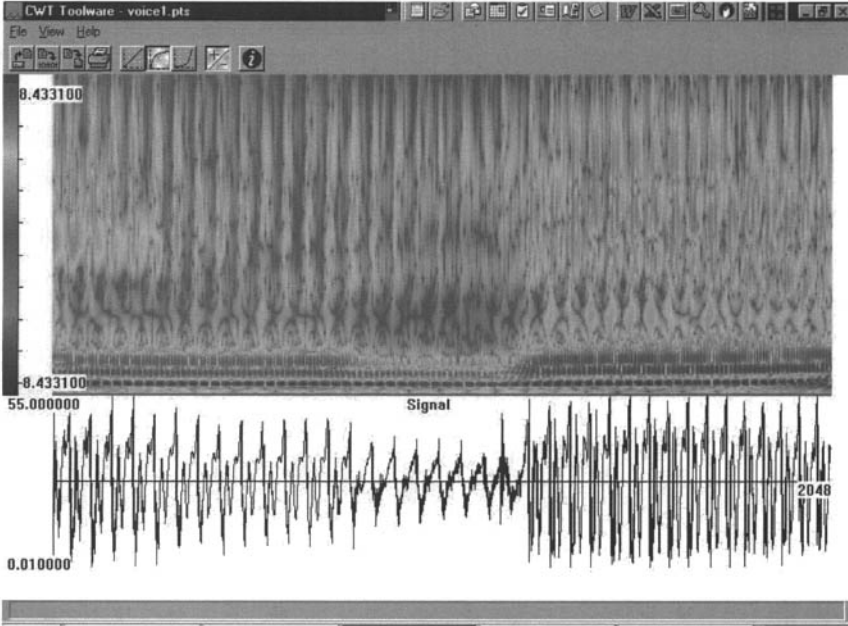


Figure 1.3: The graphical display of the CWT coefficients of a voice signal

We remark here that the basis function $w(t - b)e^{j\omega t}$ is a sinusoidal function at a single frequency modulated by the window function $w(t)$. The window widths are fixed independent of b and ω . Based on the window measures discussed earlier in this manual, the window area in the t - f plane for the wavelet at a given scale a is given by

$$[t + a(t^* - \Delta\psi), t + a(t^* + \Delta\psi)] \times \left[\frac{1}{a}(\omega^* - \Delta\hat{\psi}), \frac{1}{a}(\omega^* + \Delta\hat{\psi}) \right]. \quad (1.17)$$

We see clearly that the window area of the wavelet remains the same, namely, $4\Delta w\Delta\hat{w}$. However the time and spectral window widths are dependent on the scale a . (See Fig 1.4.) If a becomes large (i.e., at low frequency), the window width in the time domain is large while the window width in the spectral domain becomes small. This is the case in processing signals in fine resolution in the spectral domain while the time-domain resolution is coarse. For processing of a signal at low frequency, one should use wavelets with a large scale.

Although there are striking similarities between the STFT and the CWT, their results in signal processing are quite different. One must compute the STFT of a broadband signal on the t - f plane for several sizes of the same window to capture the high- and low-frequency characteristics of the signal. However, one only needs to compute the CWT once to detect both the high- and low-frequency events in the signal.

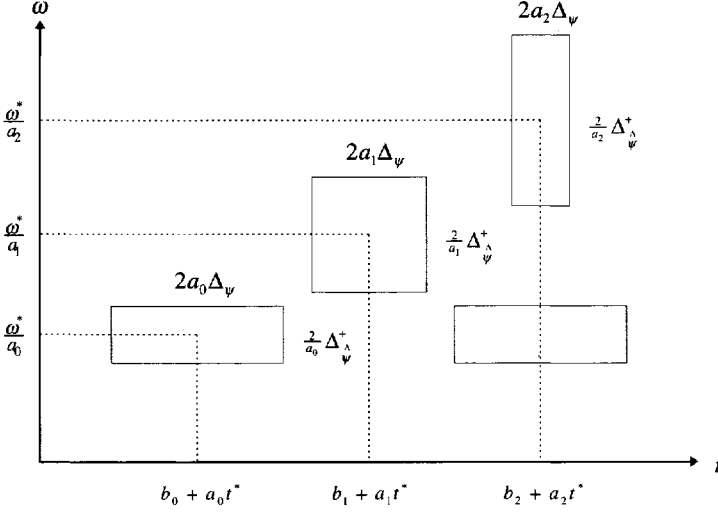


Figure 1.4: Window widths for wavelets

The original signal is uniquely recovered by the double integral

$$x(t) = \frac{1}{C_\psi} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} W_\psi(x)(b, a) \psi_{ba}(t) \frac{db da}{a^2}, \quad (1.18)$$

where C_ψ is a finite constant given by the integral

$$C_\psi = \int_{-\infty}^{\infty} \frac{|\hat{\psi}(\omega)|^2}{|\omega|} d\omega < \infty. \quad (1.19)$$

1.6 Multiresolution Analysis

We have shown that the CWT has a unique advantage because its window widths can be controlled by the scale parameter a . However, we also see that the computation load of the CWT is quite heavy in order to capture all the characteristics of the signal. To alleviate this computational burden, mathematicians have developed the Discrete Wavelet Transform (DWT) to minimize the redundancies existing in CWT. Although the algorithm of DWT is identical to that of the two-channel filter bank analysis, the underlying meanings of these algorithms are different. We will point out these differences in later sections.

The most important feature of multiresolution analysis (MRA) MRA is the ability to separate a signal into many components at different

scales (or resolutions). For a specific choice of the scaling parameter (such as $a = 2^j, j \in \mathbf{Z}$), the decomposition algorithm is equivalent to putting signal components into successive frequency octaves. Similar to multiband signal decomposition, the goal here is to apply the “divide and conquer” strategy on the signal so that individual components may be processed by different algorithms. We present here the essence of the MRA by considering the properties of the approximation subspaces and the wavelet subspaces.

Approximation subspaces

A function $\phi(t) \in L^2$, called a **scaling function** (or an approximate function), generates a nested sequence of subspaces (approximation subspaces) $\{V_j\}$ of L^2 such that

$$\{\mathbf{0}\} \longleftarrow \dots \subset V_{-2} \subset V_{-1} \subset V_0 \subset V_1 \subset V_2 \subset \dots \longrightarrow \mathbf{L}^2 \quad (1.20)$$

and satisfies a dilation (refinement) equation

$$\phi(t) = \sum_k p_k \phi(at - k) \quad (1.21)$$

for some $a > 0, a \neq 1$, and some coefficient sequence $\{p_k\} \in \ell^2$. For the DWT to be discussed in some detail later, we will consider $a = 2$ that corresponds to a dyadic octave scale in the spectral domain. More precisely, V_0 is generated by $\{\phi(\cdot - k) : k \in \mathbf{Z}\}$. In general, V_n is generated by $\{\phi(2^n \cdot - k) : k \in \mathbf{Z}\}$. The symbol $(\cdot)$ stands for the dummy variable.

Wavelet subspaces

Let the subspace V_n constitute an orthogonal sum (for which we use the symbol $\oplus$) of mutually orthogonal subspaces W_j of L^2 as

$$V_n = \oplus_{j=-\infty}^{n-1} W_j. \quad (1.22)$$

Then

$$\begin{aligned} V_{n+1} &= \oplus_{j=-\infty}^n W_j \\ &= \oplus_{j=-\infty}^{n-1} W_j \oplus W_n \\ &= V_n \oplus W_n. \end{aligned} \quad (1.23)$$

The subspace W_n (the wavelet subspace) is said to be the orthogonal complementary subspace of V_n in V_{n+1} . This relationship is best

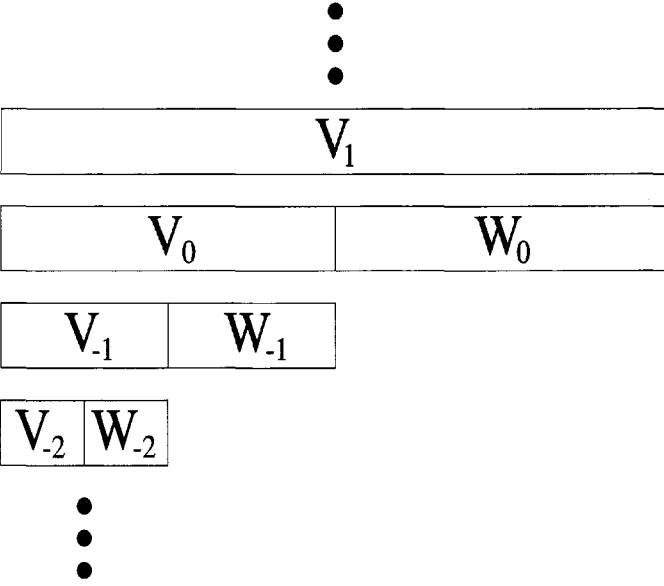


Figure 1.5: Approximation subspaces and wavelet subspaces

described by Fig 1.5. We should take note of the following two observations:

1. Subspaces $\{V_j\}$ are nested.
2. Subspaces $\{W_j\}$ are mutually orthogonal.

Mathematically, they are expressed by

$$V_j \cap V_\ell = V_j, \ell > j \quad (1.24)$$

$$W_j \cap W_\ell = \{0\}, j \neq \ell \quad (1.25)$$

$$V_j \cap W_\ell = \{0\}, j \leq \ell. \quad (1.26)$$

The subspaces $\{W_n\}$ are generated by some function $\psi(t)$, called a “wavelet,” in the same way as $\{V_n\}$ are generated by $\phi(t)$. In other words, any function $f_j(t) \in V_j$ can be written as

$$f_j(t) = \sum_{k=-\infty}^{\infty} c_{j,k} \phi(2^j t - k) = \sum_{k=-\infty}^{\infty} c_{j,k} \phi_{j,k}, \quad (1.27)$$

and any function $g_j(t) \in W_j$ can be written as

$$g_j(t) = \sum_{k=-\infty}^{\infty} d_{j,k} \psi(2^j t - k) = \sum_{k=-\infty}^{\infty} d_{j,k} \psi_{j,k}, \quad (1.28)$$

for some coefficient sets $\{c_{j,k}\}_{k \in \mathbf{Z}}$ and $\{d_{j,k}\}_{k \in \mathbf{Z}}$ in ℓ^2 . Here we have adopted the notation

$$\phi_{k,j} = \phi(2^j t - k)$$

$$\psi_{k,j} = \psi(2^j t - k).$$

We may restate here that a function $f(t)$ can be decomposed into many components such that

$$f_M(t) = \sum_{m=1}^{M'} g_{M-m}(t) + f_{M-M'}(t), \quad (1.29)$$

where the function $f_M(t)$ is the approximation of $f(t)$ from the space V_M . For example, if a signal is sampled at the rate of 256 samples per second, we may represent the signal by

$$f_8(t) = \sum_{k=-\infty}^{\infty} c_{8,k} \phi(2^8 t - k). \quad (1.30)$$

This signal can also be represented by

$$f_8(t) = \sum_{m=1}^3 g_{8-m}(t) + f_5(t). \quad (1.31)$$

1.6.1 Two-scale relations

The equation in (1.21) expresses the relation of a basis function between two different scales. The two-scale relation formalizes this relationship mathematically. Since

$$\phi(t) \in V_0 \subset V_1 \quad (1.32)$$

$$\psi(t) \in W_0 \subset V_1, \quad (1.33)$$

we can write $\phi(t)$ and $\psi(t)$ in terms of the bases that generate V_1 . In other words, there exist two sequences $\{p_k\}$ and $\{q_k\} \in \mathbf{L}^2$ such that

$$\phi(t) = \phi(2^0 t) = \sum_k p_k \phi(2^1 t - k) = \sum_k p_k \phi_{1,k}(t) \quad (1.34)$$

$$\psi(t) = \psi(2^0 t) = \sum_k q_k \phi(2^1 t - k) = \sum_k q_k \phi_{1,k}(t). \quad (1.35)$$

In general, for any $j \in \mathbf{Z}$, the relationships between V_j, W_j and V_{j+1} are governed by the two-scale relations

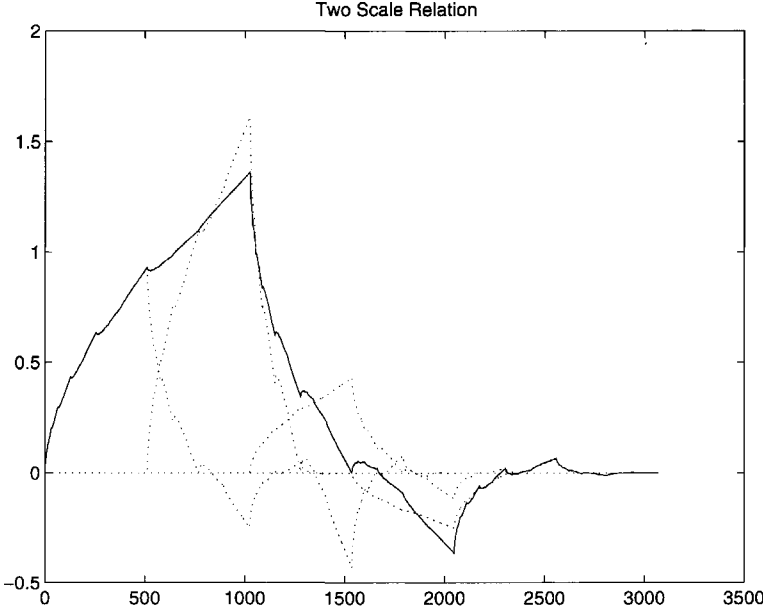


Figure 1.6: Daubechies D2 scaling function

$$\phi(2^j t) = \sum_k p_k \phi(2^{j+1} t - k) \quad (1.36)$$

$$\psi(2^j t) = \sum_k q_k \phi(2^{j+1} t - k). \quad (1.37)$$

These formulas of the basis function sets $\{\phi(2^j t)\}$ and $\{\psi(2^j t)\}$ are built up by a linear combination of $\phi(2^{j+1} t)$. For a given $\{p_k\}$ sequence, the relations (1.36) and (1.37) can be graphically describable. We use the scaling functions of Daubechies (D_2) and the cubic spline as examples. The two scale relation is graphically displayed in Fig 1.6 and Fig 1.7.

1.6.2 Decomposition relation

The relations (1.36) and (1.37) are often referred to as the reconstructive relations. The decomposition relation is for separating a signal into components at different scales. The subspace relation in an MRA

$$V_{j+1} = V_j + W_j$$

indicates that there exists a relation between the basis functions in these subspaces. Since $\phi(2t)$ and $\phi(2t - 1)$ are two distinct basis functions in

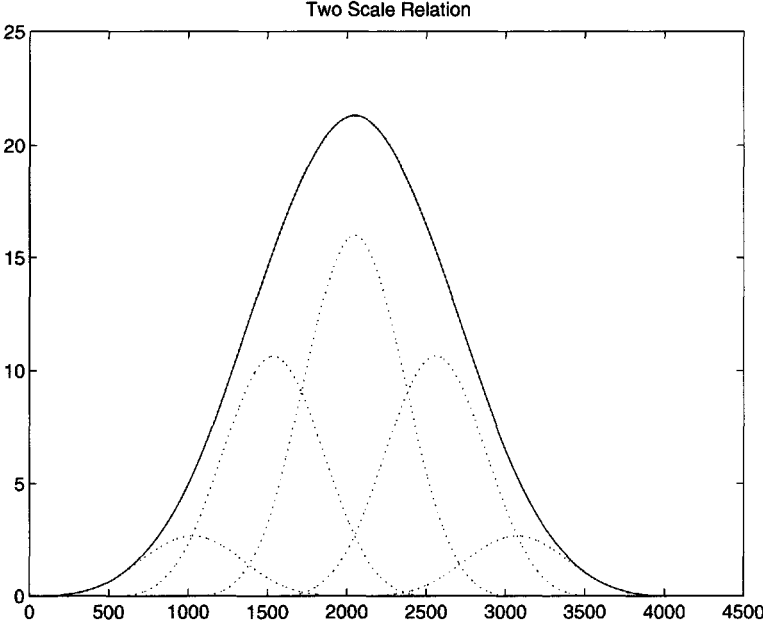


Figure 1.7: Cubic B-spline

V_{j+1} , there exist two sequences $\{a_k\}$ and $\{b_k\}$ in ℓ^2 such that

$$\begin{aligned}\phi(2t) &= \sum_k \{a_{2k}\phi(t-k) + b_{2k}\psi(t-k)\} \\ \phi(2t-1) &= \sum_k \{a_{2k-1}\phi(t-k) + b_{2k-1}\psi(t-k)\}.\end{aligned}\quad (1.38)$$

Writing these relations in their generalized form, we have

$$\phi(2t-\ell) = \sum_k \{a_{2k-\ell}\phi(t-k) + b_{2k-\ell}\psi(t-k)\} \quad (1.39)$$

for all $\ell \in \mathbf{Z}$. For an arbitrary resolution level j , this relation becomes

$$\phi_{\ell,j+1} = \phi(2^{j+1}t - \ell) \quad (1.40)$$

$$= \sum_k \{a_{2k-\ell}\phi(2^j t - k) + b_{2k-\ell}\psi(2^j t - k)\} \quad (1.41)$$

$$= \sum_k \{a_{2k-\ell}\phi_{k,j} + b_{2k-\ell}\psi_{k,j}\}. \quad (1.42)$$

There exists a definite relationship between the two-scale sequences $\{p_k\}, \{q_k\}$ and the decomposition sequences $\{a_k\}, \{b_k\}$. We will consider these relations when different types of wavelets are discussed.

1.6.3 Development of the decomposition and reconstruction algorithms

The decomposition (analysis) and reconstruction (synthesis) algorithms are the most often-used algorithms in wavelet signal processing. As mentioned earlier, the idea consists of simply dividing the signal into components so that each component may be processed with a different algorithm. The important issue of this algorithm is to be able to reconstruct the signal perfectly if we apply all-pass filters on all signal components. These algorithms are based on the two-scale relations and the decomposition relation as developed in the previous sections. We rewrite these relations here for convenience.

Let

$$\begin{aligned} f_{j+1}(t) &\in V_{j+1}, \text{ and } f_{j+1}(t) = \sum_k c_{j+1,k} \phi_{j+1,k}(t) \\ f_j(t) &\in V_j, \text{ and } f_j(t) = \sum_k c_{j,k} \phi_{j,k}(t) \\ g_j(t) &\in W_j, \text{ and } g_j(t) = \sum_k d_{j,k} \psi_{j,k}(t). \end{aligned} \quad (1.43)$$

Since the MRA requires that

$$V_{j+1} = V_j + W_j, \quad (1.44)$$

we have

$$f_{j+1}(t) = f_j(t) + g_j(t) \quad (1.45)$$

$$\sum_k c_{j+1,k} \phi_{j+1,k}(t) = \sum_k c_{j,k} \phi_{j,k}(t) + \sum_k d_{j,k} \psi_{j,k}(t). \quad (1.46)$$

We substitute the decomposition relation (1.40) in (1.46) and interchange the order of summations. Comparing the coefficients of $\phi_{j,k}(t)$ and $\psi_{j,k}(t)$, we obtain

$$c_{j,k} = \sum_{\ell} a_{2k-\ell} c_{j+1,\ell} \quad (1.47)$$

$$d_{j,k} = \sum_{\ell} b_{2k-\ell} c_{j+1,\ell}, \quad (1.48)$$

where the right sides of (1.47) and (1.48) correspond to a down-sample after convolution. These formulas relate the coefficients of the scaling function from a given scale to the scaling function and wavelet coefficients at twice the given scale.

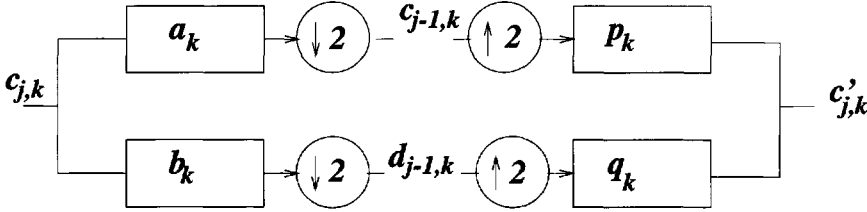


Figure 1.8: Decomposition and reconstruction

The reconstruction algorithm is based on the two-scale relations. When we sum the functions at the j th resolution, we have

$$f_j(t) + g_j(t) = \sum_k c_{j,k} \phi_{j,k}(t) + \sum_k d_{j,k} \psi_{j,k}(t) = f_{j+1}(t). \quad (1.49)$$

We substitute the two-scale relations into (1.49) to yield

$$\begin{aligned} & \sum_k c_{j,k} \sum_\ell p_\ell \phi_{j+1,k}(t - \ell) + \sum_k d_{j,k} \sum_\ell q_\ell \phi_{j+1,k}(t - \ell) \\ &= \sum_k c_{j+1,k} \phi_{j+1,k}(t). \end{aligned} \quad (1.50)$$

Again, we compare the coefficients of $\phi_{j+1,k}(t)$ on both sides of (1.50) to yield

$$c_{j+1,\ell} = \sum_k [p_{\ell-2k} c_{j,k} + q_{\ell-2k} d_{j,k}] \quad (1.51)$$

where the right sides of the equations correspond to an up-sampling before convolution. We use a block diagram (Fig 1.8) to illustrate the procedure of these algorithms.

1.6.4 Mapping of functions into the approximation space

Before we go on any further on the extensions of the decomposition and reconstruction algorithms, we first need to show how to obtain the scaling function representation of an input data. That is, for a given input signal $s(t)$, we wish to find the representation

$$s(t) \cong s_j(t) = \sum_k c_{j,k} \phi_{j,k}(t) = \sum_k c_{j,k} \phi(2^j t - k), \quad (1.52)$$

where $c_{j,k}$ are the input scaling function coefficients to be processed. Many users of wavelets have used the sample (discretized) values of $s(t = \frac{k}{2^j})$ as the input. We want to stress the fact that in general,

$$s(t = \frac{k}{2^j}) \neq c_{j,k}. \quad (1.53)$$

If we require $s(\frac{k}{2^j}) = s_j(\frac{k}{2^j})$, then the two sides of (1.53) are equal to each other *only* if the scaling function is interpolatory, such as first-order B-spline that corresponds to the Haar wavelet and the linear B-spline. It is therefore advisable to obtain $c_{j,k}$ from the sample data $s(t = \frac{k}{2^j})$ before applying the decomposition algorithm. In the following, we consider the orthogonal projection of $s(t)$ onto the V_j space.

Assuming V_j is a subspace of L^2 and $s(t) \in L^2$, we wish to find $s_j(t) \in V_j$ as an approximation to $s(t)$. Write

$$s_j(t) = \sum_k c_{j,k} \phi_{j,k}(t) \quad (1.54)$$

and consider $s_j(t)$ to be the orthogonal projection of $s(t)$ onto the V_j subspace. Then, $s(t) - s_j(t)$ is orthogonal to V_j so that

$$\langle s(t) - s_j(t), \phi_{j,\ell} \rangle = 0 \quad (1.55)$$

$$\langle s(t), \phi_{j,\ell} \rangle = \langle \sum_k c_{j,k} \phi_{j,k}(t), \phi_{j,\ell}(t) \rangle. \quad (1.56)$$

In matrix form this equation becomes

$$\begin{bmatrix} \cdot & & & & \\ & \cdot & \cdot & & \\ & a_1 & a_0 & a_1 & \\ & & \cdot & a_0 & a_1 \\ & & & \cdot & a_0 \\ & & & & \cdot \end{bmatrix} \begin{bmatrix} \cdot \\ \cdot \\ c_{j,k} \\ \cdot \\ \cdot \\ \cdot \end{bmatrix} = 2^j \begin{bmatrix} \cdot \\ \cdot \\ \langle s, \phi_{j,k} \rangle \\ \cdot \\ \cdot \\ \cdot \end{bmatrix}, \quad (1.57)$$

where

$$a_m = \int_{-\infty}^{\infty} \phi(t) \overline{\phi(t-m)} dt \quad (1.58)$$

is the autocorrection sequence of the scaling function. If the scaling function is compactly supported, the corresponding autocorrelation matrix in (1.57) is banded with a finite number of diagonal bands. Also, if the scaling function forms an orthonormal basis, then

$$a_m = \delta_{m,0}. \quad (1.59)$$

If we have only the sample values of the signal $s \Rightarrow s(t = \frac{k}{2^j})$, we have to approximate the integral in (1.56) by a sum

$$\begin{aligned} c_{j,\ell} &= \langle s(t), \phi_{j,\ell} \rangle \\ &= \int_{-\infty}^{\infty} s(t) \overline{\phi(2^j t - \ell)} dt \\ &\cong \sum_k s(t = \frac{k}{2^j}) \overline{\phi(k - \ell)}. \end{aligned} \quad (1.60)$$

This demonstrates the difference between the coefficients of the scaling function series and the sample values of the signal. The former is used in wavelet signal processing (analog processing in the time domain, since the coefficient sequence of the scaling function series and the wavelet series are to be processed), while the latter is used in filter bank processing (digital processing in the frequency domain).

1.7 Types of Wavelets

In the preceding discussions of the approximate subspace and wavelet subspace, the basis functions that span the subspace can have orthonormal, semi-orthogonal, or biorthogonal bases. There are conditions to be satisfied by each type of basis. We will state these conditions separately as follows.

1.7.1 Orthonormal wavelet bases

Orthogonality of functions depends on the definition of the inner product in that vector space. We have used the inner product defined in the L^2 space. Two different functions are orthogonal to each other if the inner product of these functions is zero. That is,

$$\langle f, g \rangle = \int_{-\infty}^{\infty} f(x) \overline{g(x)} dx = 0. \quad (1.61)$$

If we take $g(x) = f(x)$, and

$$\langle f, f \rangle = \int_{-\infty}^{\infty} f(x) \overline{f(x)} dx = 1, \quad (1.62)$$

we say that the function $f(x)$ is normalized. Hence, if the set of basis functions $\phi_j(x)$, $j \in Z$ spans an approximation space V_j and satisfies the condition

$$\langle \phi_i, \phi_j \rangle = \int_{-\infty}^{\infty} \phi(x-i) \overline{\phi(x-j)} dx = \delta_{i,j}, \quad (1.63)$$

it is a normalized orthogonal set, or, for short, an orthonormal set. This definition applies to the wavelet subspace as well. A wavelet is said to be orthonormal if the set of basis functions $\psi_{j,k} = 2^{j/2} \psi(2^j x - k)$ satisfies

$$\langle \psi_{i,\ell}, \psi_{j,k} \rangle = \int_{-\infty}^{\infty} \psi_{i,\ell}(x) \overline{\psi_{j,k}(x)} dx = \delta_{i,j} \delta_{\ell,k}. \quad (1.64)$$

Since the wavelet subspace and the approximation subspace are orthogonal, we have

$$\langle \phi_{0,i}, \psi_{0,j} \rangle = \int_{-\infty}^{\infty} \phi_{0,i}(x) \overline{\psi_{0,j}(x)} dx = 0, \quad \forall i, j \in \mathbf{Z}. \quad (1.65)$$

Notice that the relationships in (1.63) to (1.65) are all referred to the same level of resolution. The relationships between wavelets at different resolution levels are also orthogonal. That is,

$$\langle \psi_{k,i}, \psi_{\ell,j} \rangle = \delta_{i,j} \delta_{k,\ell}. \quad (1.66)$$

Examples of the orthonormal wavelet basis include the Haar wavelets, the Daubechies orthonormal wavelet bases of all orders, the Meyer wavelets of all orders, and Battle–Lemarie orthonormal spline wavelets (not compactly supported). We note here that both Meyer and Battle–Lemarie wavelets are not compactly supported (finite time duration).

Decomposition and reconstruction sequences

The relations between the decomposition sequences $\{a_k\}$, $\{b_k\}$ and the reconstruction sequences $\{p_k\}$, $\{q_k\}$ are the simplest for an orthonormal basis. For perfect reconstruction of the signal, we may use the following relations:

$$\begin{aligned} q_k &= (-1)^k p_{1-k} \\ a_k &= -p_k \\ b_k &= -q_k. \end{aligned}$$

Once the p_k sequence has been found through the construction of the two-scale relation of the scaling function, the other sequences are determined.

1.7.2 Semi-orthogonal wavelet bases

In the case of cardinal B-spline functions of orders higher than 1 and their corresponding compactly supported spline wavelets, the integer translates of the scaling functions, i.e., B-splines, as well as the B-wavelets are not orthogonal. That is,

$$\int_{-\infty}^{\infty} N_m(x-i) \overline{N_m(x-j)} dx \neq \delta_{i,j} \quad (1.67)$$

$$\int_{-\infty}^{\infty} \psi_{N_m}(x-i) \overline{\psi_{N_m}(x-j)} dx \neq \delta_{i,j}. \quad (1.68)$$

However, the orthogonality between the scaling function N_m and the B-wavelet ψ_{N_m} , as well as translates of wavelets at different resolutions, is still preserved. In this case, the compactly supported spline wavelet is called a semi-orthogonal wavelet. In order to facilitate efficient computations of the expansion coefficients in the approximation spaces or the wavelet spaces, a dual scaling function $\tilde{\phi}$ and a dual wavelet $\tilde{\psi}$ can be found to satisfy the biorthogonality relation

$$\langle \phi_{k,i}, \tilde{\phi}_{\ell,j} \rangle = \delta_{i,j} \delta_{k,\ell} \quad (1.69)$$

$$\langle \psi_{k,i}, \tilde{\psi}_{\ell,j} \rangle = \delta_{i,j} \delta_{k,\ell}, \quad (1.70)$$

where $\phi = N_m$ and $\psi = \psi_{N_m}$ have been used in these two equations. We remark here that both the dual scaling function (the dual B-spline) and the dual spline wavelet are both functions in the spline space of the same order. Hence, these duals can be expanded into B-spline series of the same spline order. We will show in later sections how to use these relations to map coefficients between wavelet spaces and dual wavelet spaces.

Decomposition and reconstruction sequences

The sequences for the two-scale relations in the m^{th} order spline space are

$$p_k^m = \begin{cases} \frac{1}{2^{m-1}} \binom{m}{\ell}, & 0 \leq k \leq m \\ 0, & \text{otherwise} \end{cases}$$

$$q_k^m = \frac{(-1)^k}{2^{m-1}} \sum_{\ell=0}^m \binom{m}{\ell} N_{2m}(k+1-\ell), \quad k = 0, \dots, 3m-2.$$

The two-scale relations for the dual spline and dual wavelets are slightly more complicated. Since the B-spline functions of order $m \geq 2$ are not orthogonal basis, the sequences $\{a_k\}, \{b_k\}$ comes from the dual spline and dual wavelets which are not compactly supported, but they are both infinite sequences with exponential decay. Explicit expressions for these sequences are

$$\begin{aligned} a_k &= 2^{-m+1} \sum_{\ell=0}^m \binom{m}{\ell} \sum_{2h=k-\ell-m+1}^{k-\ell+m-1} N_{2m}(m+k-2h-\ell) g_h; \\ b_k &= (-1)^{k+1} 2^{-m+1} \sum_{2h=k-2m}^{k-m} \binom{m}{k-2h-m} g_h, \end{aligned}$$

where the z -transform of $\{g_h\}$ is the reciprocal of the z -transform of the Euler-Frobenius-Laurent Polynomial $E_\phi(z)$, namely

$$\begin{aligned} G(z) &= \sum_h g_h z^h := \frac{1}{E_\phi(z)} \\ &= \left[\sum_{h=-m+1}^{m-1} N_{2m}(m+h) z^h \right]^{-1} \end{aligned}$$

Numerical values for these sequences are listed in Table 5.5 of the book *Wavelets: A Mathematical Tool for Signal Analysis* (SIAM Publ., 1997) for $m = 2$ (linear splines) and $m = 4$ (cubic splines).

1.7.3 Biorthogonal wavelet bases

It is known in the signal processing literature that finite impulse response (FIR) filters with symmetric or antisymmetric coefficients have linear phase or psuedo-linear phase characteristics. Linear phase properties are important for minimizing signal distortion during processing. The processing sequences (or the FIR filters) $\{p_k\}$ and $\{q_k\}$ of finitely supported orthogonal wavelets are generally not symmetric. In order to take advantage of the symmetric property, one has to give up orthogonality. Filter designs in a two-channel filter banks are examples of a biorthogonal basis. In this case, orthogonality between filters is not required. The only requirement for filter bank design is that the output of the filter bank be a delayed version of the input. That is, the output signal must be a perfect copy (perfect reconstruction condition) of the input signal. Biorthogonal wavelets of Cohen, Daubechies, and Feauvears are examples of this class of bases. In addition, there may not exist a scaling function or wavelet associated with filters designed for filter bank processing. In other words, there may not be a nested V spaces for arbitrary FIR filters as their two-scale sequences. For this reason, the two-channel filter bank is an excellent alternative for processing digital signals.

Decomposition and reconstruction sequences

The two-scale sequences of two biorthogonal wavelets of Daubechies *et al.* are listed here as a reference. For more details of biorthogonal wavelets, readers are referred to Section 5.6 (particularly, Example 5.41) of the accompanying text *An Introduction to Wavelets* [3]; the text *Wavelets: A Mathematical Tool for Signal Analysis*, Chapter 5, by

Chui [2]; and the engineering texts, *Wavelets and Subband Coding* by Vetterli and Kovacevic, Prentice Hall, 1995 [4], and *Wavelets and Filter Banks* by Strang and Nguyen, Wellesley-Cambridge Press, 1996 [5]. Coefficients of the B-97 Daubechies biorthogonal wavelet are given in the following table.

BiDaub9.7

k	a_k	b_k	p_k	q_k
-5				
-4	0.026748757411			
-3	-0.016864118443			
-2	-0.078223266529	-0.045635881557	-0.091271763114	-0.053497514822
-1	0.266864118443	0.028771763114	-0.057543526228	-0.033728236886
0	0.602949018236	0.295635881557	0.591271763114	0.156446533058
1	0.266864118443	0.295635881557	1.115087052458	0.533728236886
2	-0.078223266529	-0.557543526229	0.591271763114	-1.205898036472
3	-0.016864118443	0.295635881557	-0.057543526228	0.533728236886
4	0.026748757411	0.028771763114	-0.091271763114	0.156446533058
5		-0.045635881557		-0.033728236886
				-0.053497514822

1.8 Wavelet Packets

The hierarchical wavelet decomposition produces signal components whose spectra reside in consecutive octave bands. In certain applications, the spectral resolution produced by this decomposition may not be fine enough. One may want to use the CWT to obtain the necessary finer resolutions by changing the scale parameter a by a small increment. However, this increases the computation load by orders of magnitude. In order to avoid this problem, wavelet packets are generalizations of wavelets in that each octave frequency band is itself further subdivided into finer frequency bands by wavelet packet transforms. In other words, the development of wavelet packets is a refinement of wavelets in the frequency domain and is based on a mathematical theorem proven by I. Daubechies (the) splitting trick). The theorem is stated as follows:

If $f(\cdot - k) \mid_{k \in \mathbb{Z}}$ forms an orthonormal basis and

$$F_1(x) = \sum_k p_k f(x - k) \quad (1.71)$$

$$F_2(x) = \sum_k q_k f(x - k), \quad (1.72)$$

then $\{F_1(\cdot - 2k), F_2(\cdot - 2k); k \in \mathbb{Z}\}$ is an orthonormal basis for $E =$

$\text{span}\{f(\cdot - n); n \in \mathbf{Z}\}$. The proof of this theorem is given in her book *Ten Lectures on Wavelets* (SIAM Publ., 1992, p. 326) [6].

This theorem is obviously true when f is the scaling function ϕ , since the two-scale relations for ϕ and the wavelet ψ give

$$\begin{aligned} V_j &\ni \phi(2^j t) = \sum_k p_k \phi(2^{j+1} t - k) \\ W_j &\ni \psi(2^j t) = \sum_k q_k \phi(2^{j+1} t - k). \end{aligned}$$

If we apply this theorem to the W_j spaces, we generate the wavelet packet subspaces. The general recursive formulas for wavelet packet generation are

$$\mu_{2\ell}(t) = \sum_k p_k \mu_\ell(2t - k) \quad (1.73)$$

$$\mu_{2\ell+1}(t) = \sum_k q_k \mu_\ell(2t - k) \quad k \in \mathbf{Z}, \quad (1.74)$$

where $\mu_0 = \phi$ and $\mu_1 = \psi$ are the scaling function and the wavelet, respectively. For $\ell = 1$, we have the wavelet packets μ_2 and μ_3 generated by the wavelet $\mu_1 = \psi$. This process is repeated so that many wavelet packets can be generated from the two-scale relations. Because of the many components available, any given signal can be represented by a choice of wavelet packets at different levels of resolution. An optimized algorithm can be constructed to maximize a certain measure (such as energy or entropy). Best-bases and best-level are two popular algorithms for signal representations. The reader can find these algorithms in [7]. The Toolware is designed so that the decomposition and reconstruction algorithms can use either wavelets or wavelet packets.

1.8.1 Wavelet packet algorithms

The wavelet packet algorithm follows the wavelet algorithm with the extension that the wavelet coefficients $d_{j,k}$ are also being processed in the same way as the scaling function coefficients $c_{j,k}$. Hence, for every decomposition process, the number of components is double that of the previous resolution. Suppose an input signal $s(n)$ has been mapped into the approximation space V_j . That is, $s(n) \longleftrightarrow c_{j,k}$. Using the decomposition sequences a_n and b_n , we obtain the scaling function series coefficients $c_{j-1,k}$ and the wavelet series coefficients $d_{j-1,k}$. We can represent the scaling function component and the wavelet component at

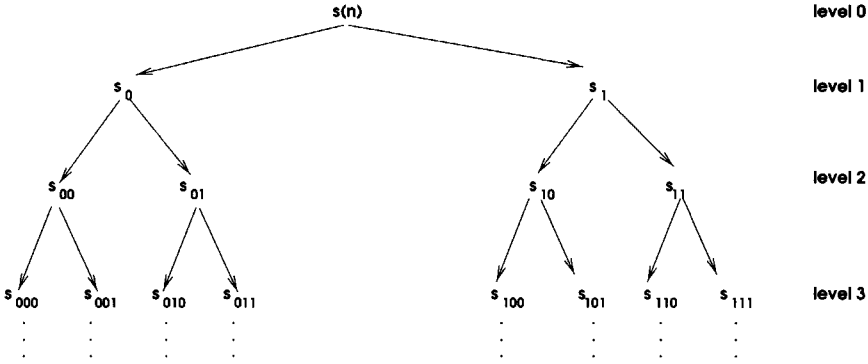


Figure 1.9: Wavelet packet decomposition tree

resolution $j - 1$,

$$s_0(k) \longleftrightarrow c_{j-1,k} \quad (1.75)$$

$$s_1(k) \longleftrightarrow d_{j-1,k}. \quad (1.76)$$

The coefficient sequences $c_{j-1,k}$ and $d_{j-1,k}$ are decomposed through a_n and b_n into four sequences

$$\begin{aligned} s_{00}(k) &\longleftrightarrow c_{j-2,k} \\ s_{01}(k) &\longleftrightarrow d_{j-2,k} \\ s_{10}(k) &\longleftrightarrow u_{j-2,k} \\ s_{11}(k) &\longleftrightarrow v_{j-2,k} \end{aligned} \quad (1.77)$$

in which the last two sequences $u_{j-2,k}$ and $v_{j-2,k}$ are wavelet packet series coefficients. This process can be repeated for the four sequences in (1.77) to produce eight coefficient sequences corresponding to eight wavelet packet components of the original signal $s(n)$. The decomposition tree structure is given in Fig 1.9. The algorithm for WP decomposition and reconstruction of a one-dimensional signal is shown in Fig 1.10.

1.9 Two-Dimensional Wavelets

When the input signal is two-dimensional (2-D), it is necessary to represent the signal components by two-dimensional wavelets and two-dimensional scaling functions. For any scaling function ϕ with its corresponding wavelet ψ , there are three different 2-D wavelets and one 2-D scaling function using the tensor-product approach. We can write

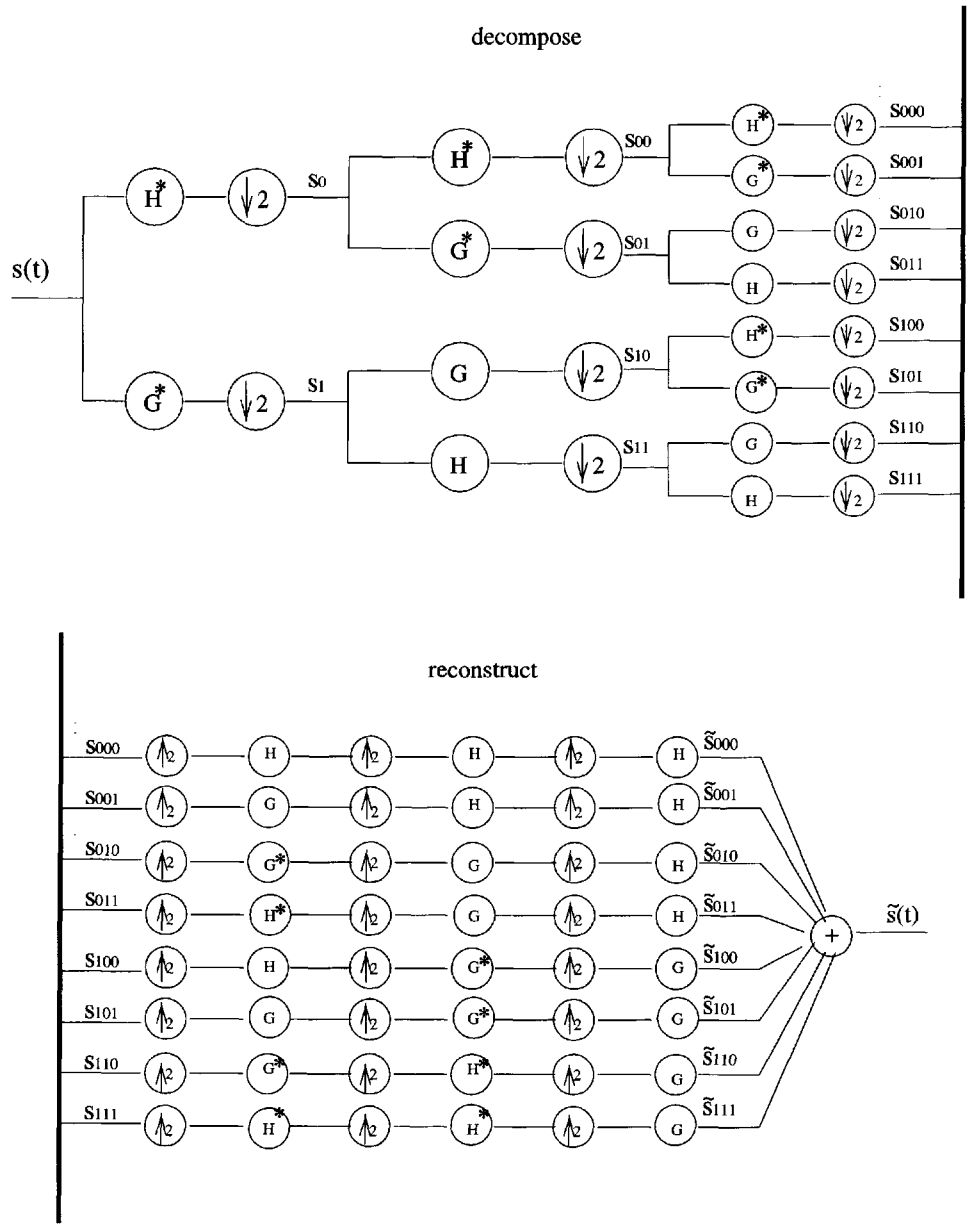


Figure 1.10: Block diagram for the WP decomposition of a signal

the 2-D wavelets as

$$\Psi_{i,j}^{[1]}(x, y) = \phi(x - i)\psi(y - j) \quad (1.78)$$

$$\Psi_{i,j}^{[2]}(x, y) = \psi(x - i)\phi(y - j) \quad (1.79)$$

$$\Psi_{i,j}^{[3]}(x, y) = \psi(x - i)\psi(y - j), \quad (1.80)$$

and the 2-D scaling function as

$$\Phi_{i,j}(x, y) = \phi(x - i)\phi(y - j). \quad (1.81)$$

In the spectral domain, each of the wavelets and the scaling function occupies a different portion of the 2-D spectral plane. When we analyze a 2-D signal, the decomposition algorithm generates four components from the input signal. With respect to the spectral domain, the components are labeled low-high (LH), high-low (HL), and high-high (HH), corresponding to the wavelets $\Psi_{i,j}^{[M]}(x, y)$, $M = 1, 2, 3$. The component that corresponds to the scaling function signal is called the low-low (LL) component. The terms low and high refer to whether the processing filter is lowpass or highpass. The decomposition of a 2-D signal results in what is known as a hierarchacal pyramid. Because of the downsampling operation, each image is decomposed into four subimages. The size of each subimage is only a quarter of that of the original image. Readers can refer to the last section of this manual to try to decompose a 2-D image signal and see the resultant component display.

1.9.1 Two-dimensional wavelet packets

Two-dimensional wavelet packets are refinements of 2-D wavelets just as in the 1-D case. Not only is the LL component (the scaling function component) decomposed to obtain further details of the image, but the other wavelet components (LH, HL, HH) can also be further decomposed. For example, starting with an original image with size 128×128 , a 2-D wavelet decomposition of this image will result in four subimages of size 64×64 . Continuing the decomposition, one gets 16 2-D wavelet packet subimages of size 32×32 . The computational algorithm for 2-D wavelet packets is no different than that for the 2-D wavelets. It requires an orderly bookkeeping for keeping track of the directions (x or y), the filters used (highpass or lowpass), and the resolutions. The reader can make use of this wavelet Toolware to practice the wavelet packet decomposition and reconstruction of 2-D signals.

Part II

Wavelet Algorithms

This page intentionally left blank

2.1 Wavelet Algorithms Overview

This part discusses the approach we take to numerically implement the mathematical statements given in Part I. In addition, we also include several algorithms that have been used in many wavelet signal processing applications such as thresholding, noise removal, and signal compression. The algorithms are written in procedural form so that the reader can easily follow them to build codes using his/her preferred computer languages. The following algorithms are included in this section:

1. Computation of STFT for 1-D signal
2. Computation of CWT for 1-D signal
3. Algorithm for generation of the graph of scaling functions
4. Algorithm for generation of the graph of wavelets
5. Mapping between B-splines and their duals
6. Decomposition of 1-D signals
7. Reconstruction of 1-D signals
8. Thresholding
9. Two-dimensional extension of the wavelet algorithms
10. Other algorithms

2.2 Algorithm for Computing the STFT

Let us rewrite the definition of STFT here for easy reference:

$$ST_w f(t, \omega) = \int_{-\infty}^{\infty} f(\tau) \overline{w(\tau - t)} e^{-j\omega\tau} d\tau.$$

A straightforward approach to computing the STFT is to make use of the Fast Fourier Transform (FFT) algorithm that can be found in many numerical analysis books. The procedure for computing the STFT is given here:

1. Choose a symmetric window function and sample it at the same rate as the input data.
2. Count the total number of samples of the window function, say N .

3. To take care of the edge (or boundary) effect of the algorithm, we may reflect $\frac{N}{2}$ (if N is even) or $\frac{N-1}{2}$ (if N is odd) data points to the outside the edges of the 1-D data set at both ends of the set. That is, if the data set has M data points, the modified data set will have a total of $M + N$ points from $-\frac{N}{2}$ to $M + \frac{N}{2}$.
4. Position the center of the window sample data (or the center of the window) at $n = 0$ and multiply the window sample with the data set. Take an N -point FFT to obtain the spectrum of the function at $n = 0$.
5. Shift the window data to the right by one point and repeat step 4 to obtain the spectrum of the function at $n = 1$.
6. Repeat step 5 until the center of the window has reached the end point of the data set at $n = M - 1$. The resulting output data should be a rectangular array of $M \times N$ points.
7. Display the data through either a 3-D plot or a gray-scale plot.

Example

We use an example similar to the one given in Dauchies' book for illustration. Let the function to be processed be

$$f(t) = \sin 1000\pi t + \sin 2000\pi t + \alpha[\delta(t - t_1) + \delta(t - t_2)], \quad (2.82)$$

where α is a constant chosen to be 3 and t_1 and t_2 are 0.192 and 0.196 msec, respectively. The function is sampled at 8000 points per second. The frequencies are chosen so that when the time window width is small enough to resolve the delta functions, the corresponding frequency window width is too wide to resolve the two frequencies accurately. We use a rectangular window of different window widths, 8 and 64 points, respectively. The STFTs of the functions for these windows are computed using the Toolware, and the results are shown in Fig 2.1 and Fig 2.2. One can easily see from these graphs that if the window in the time-domain is wide enough to resolve the low frequencies, the delta functions are not resolvable. If the time-domain window is narrow enough to resolve the delta functions, the frequency resolution is low.

We will show later that the CWT can resolve the delta functions and the frequencies simultaneously in only one pass by varying the scale parameter.

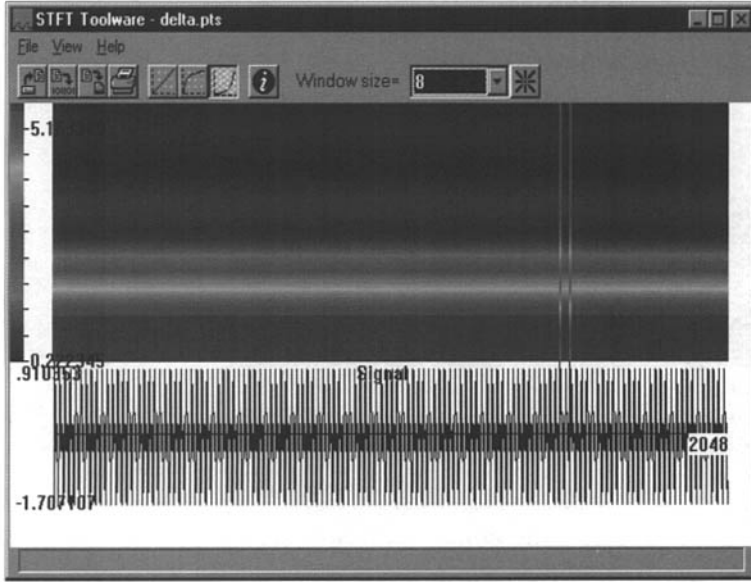


Figure 2.1: STFT outputs for the window size equal 8

2.3 Algorithm for Computing the CWT

The definition of the CWT given in part A is written here:

$$\begin{aligned}
 W_{\psi_{ba}} x(b, a) &= \frac{1}{\sqrt{a}} \int_{-\infty}^{\infty} x(t) \overline{\psi\left(\frac{t-b}{a}\right)} dt \\
 &= \int_{-\infty}^{\infty} x(t) h(b-t) dt \\
 &= x(t) * h(t),
 \end{aligned} \tag{2.83}$$

where $h(t) = \frac{1}{\sqrt{a}} \overline{\psi\left(-\frac{t}{a}\right)}$ is used in (2.83). Using the convolution integral expression, (2.83) can be computed by using FFT as in the case of STFT. The steps are outlined as follows:

1. Select a wavelet of your choice to be used for this computation.
2. For a given scale a , sample the wavelet at the same rate as the input data.
3. Treat the data at the edge the same way as indicated in the STFT.
4. Compute the FFT of the input data and that of the wavelet at scale a .
5. Multiply the two FFTs and take the IFFT to obtain the CWT of the function at the scale a .

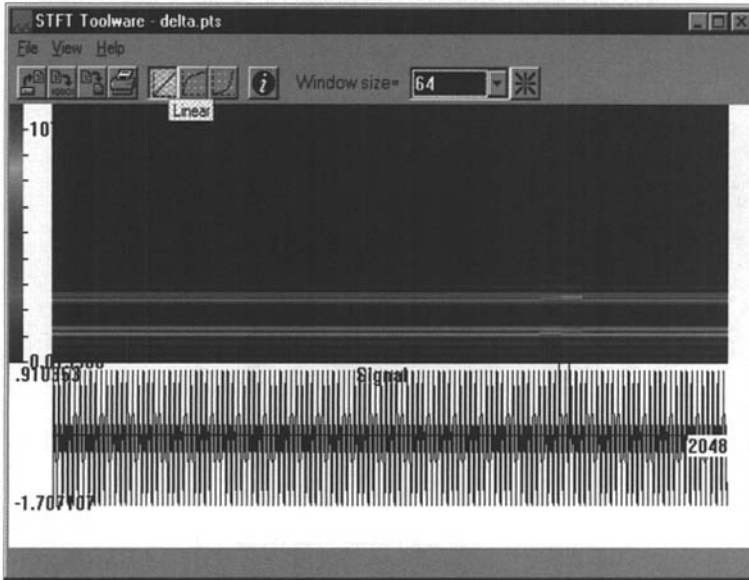


Figure 2.2: STFT outputs for the window size equal 64

6. Repeat steps 2 through 5 for as many values of a as the user wishes.
7. Display data using a 3-D plot or gray-level display.

Example

The same function used to demonstrate the STFT is also used here to demonstrate the characteristics of the CWT. We choose the Morlet wavelet and compute the CWT via FFT. The result shown in Fig 2.3 clearly shows the advantage of a flexible window of the CWT. It requires only one computation to resolve both the frequencies and the delta functions in (2.82). One can compare this graph of the CWT to that of the STFT.

2.4 Computation and Display of Scaling Functions

We give two algorithms for computing the scaling functions.

Spectral-domain method

Let us recall the two-scale relations for the scaling function

$$\phi(t) = \sum_k p_k \phi(2t - k)$$

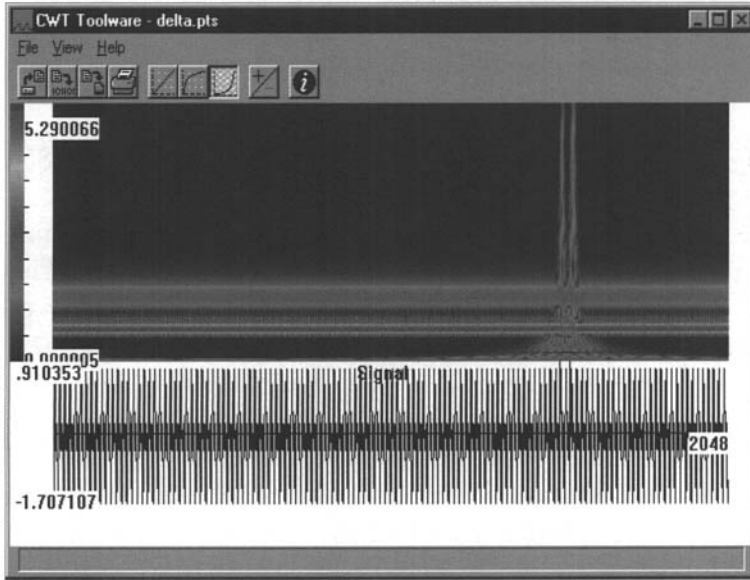


Figure 2.3: CWT outputs

and consider the spectral domain expression of this relation, namely,

$$\begin{aligned}
 \hat{\phi}(\omega) &= \frac{1}{2} \sum_k p_k \hat{\phi}\left(\frac{\omega}{2}\right) e^{-j\frac{\omega}{2}} \\
 &= P\left(e^{-j\frac{\omega}{2}}\right) \hat{\phi}\left(\frac{\omega}{2}\right) \\
 &= P\left(e^{-j\frac{\omega}{2}}\right) P\left(e^{-j\frac{\omega}{2^2}}\right) \hat{\phi}\left(\frac{\omega}{2^2}\right) \\
 &\quad \cdot \\
 &\quad \cdot \\
 &= \prod_{k=1}^N P\left(e^{-j\frac{\omega}{2^k}}\right) \hat{\phi}\left(\frac{\omega}{2^N}\right) \\
 &= \prod_{k=1}^{\infty} P\left(e^{-j\frac{\omega}{2^k}}\right), \tag{2.84}
 \end{aligned}$$

where $P(*)$ is the z -transform of the sequence p_k . Hence, the scaling function $\phi(t)$ can be obtained by finding the IFFT of the infinite product given by (2.84). In general, the infinite product must be truncated so that there is only a finite number of points in the IFFT program.

This approach is not easy to implement because the complexity of the product of the discrete Fourier transform of the sequence $\{p_k\}$. If the number of elements in this sequence is small, the problem is still manageable; otherwise, the data storage required is substantial. The following alternative approach can be used.

Iterative method

We modify the two-scale equation to fit an iterative technique by considering

$$\phi_{n+1}(x) = \sum_k p_k \phi_n(2x - k), \quad n = 0, 1, 2, \dots \quad (2.85)$$

The index n indicates the number of iterative loops. For initialization, the user may use either first- or second-order B-splines. The iteration should converge after 5 to 10 iterations if the regularity of the function is high. However, for some sequences designed by the perfect reconstruction filter bank approach, the algorithm may not even converge at all. In other words, there are filter banks that are not associated with scaling functions and wavelets. The final data may be seen by using the usual 1-D graphic display. The code to generate the scaling function in this Toolware is based on this algorithm. We outline the procedure as follows:

1. Initialize the program by setting all data files to zero.
2. Set desired number of iteration, say 10, and set the iteration index to 1.
3. Input the initial trial function $\phi_0(x)$. One may use an impulse function, a rectangular pulse (i.e., first-order B-spline), a triangular pulse (i.e., second-order B-spline), etc.
4. Carry out (3.13) by convolving the function with the p_k sequence.
5. Upsample the resulting sequence by inserting zeros in between every other data point. This sequence is $\phi_1(x)$.
6. Increase the iteration index by 1 and repeat steps 4 to 6 until 10 iteration cycles have been completed.

The reader may use the Toolware to familiarize himself/herself with scaling functions included in the software.

2.5 Computation and Graphical Display of Wavelets

To generate a wavelet that satisfies the two-scale relation

$$\psi(t) = \sum_k q_k \phi(2t - k), \quad (2.86)$$

we can use either the spectral or the time-domain approach. For the spectral-domain approach, we simply multiply the infinite product in

(2.84), except that the index k runs from 2 to ∞ , by one-half of the z -transform of $\{q_k\}$, namely,

$$\begin{aligned}\widehat{\psi}(\omega) &= \left(\frac{1}{2} \sum_k q_k e^{-j\frac{\omega}{2}} \right) \widehat{\phi}\left(\frac{\omega}{2}\right) \\ &= \left(\frac{1}{2} \sum_k q_k e^{-j\frac{\omega}{2}} \right) \Pi_{k=2}^{\infty} P\left(e^{-j\frac{\omega}{2^k}}\right).\end{aligned}$$

For the time-domain approach, we take the linear combination of the translate scaling function $\phi(2t - k)$ using the coefficients q_k . We use the time-domain approach in this Toolware to generate the graphs of different wavelets. The reader should use the Toolware to view different wavelets as given in Part C.

2.6 Mapping between B-Splines and Their Duals

When B-splines are chosen to represent a signal $s(t)$, we have shown in Part I that it is necessary to map the sample values $s\left(t = \frac{k}{2^j}\right)$ to an approximation of the signal $s_j(t) = \sum_k c_{k,j} \phi_{k,j}(t)$. This procedure maps the signal into the B-spline (or the approximation) space at the j th resolution. Since the decomposition sequences $\{a_k\}$ and $\{b_k\}$ for B-splines and B-spline wavelets are infinitely long, a truncation of the sequence is necessary with a reasonable filter length to ensure adequate accuracy. For processing of large images (e.g., mammograms at $4000 \times 5000 \times 12$ bits per image) or 3-D images (e.g., multislice CAT scans), the decomposition process is inefficient and the computational load is large. Since the B-splines $N_m(2^j t - k)$ and its duals $\widetilde{N}_m(2^j t - k)$ span the same subspace V_j , it is convenient and efficient to map the B-spline coefficients $c_{j,k}$ to the dual-spline coefficients $\widetilde{c}_{j,k}$. After that, we may process the signal using sequences $\{p_k\}$ and $\{q_k\}$, which are finite with relatively short filter lengths.

Let

$$s_j(t) = \sum_k c_{k,j} \phi_{k,j}(t) \quad (2.87)$$

$$= \sum_k \widetilde{c}_{k,j} \widetilde{\phi}_{k,j}(t) \quad (2.88)$$

be a signal approximated by a B-spline series ($\phi_{k,j}(t) = N_m(2^j t - k)$) and the dual B-spline series ($\widetilde{\phi}_{k,j}(t) = \widetilde{N}_m(2^j t - k)$), respectively. $c_{k,j}$ are the spline coefficients that have been determined through the initial

mapping from the sampled values. To obtain the dual B-spline coefficients, one simply makes use of the orthogonality condition between the B-spline and its dual,

$$\langle N_m(2^j t - \ell), \widetilde{N}_m(2^j t - k) \rangle = \delta_{k,\ell}. \quad (2.89)$$

Let $j = 0$ by making the sampling interval unity. We compute the dual coefficients $\widetilde{c}_{k,j}$ by

$$\widetilde{c}_\ell = \left\langle \sum_k \widetilde{c}_k \widetilde{N}_m(t - k), N_m(t - \ell) \right\rangle \quad (2.90)$$

$$= \left\langle \sum_k c_k N_m(t - k), N_m(t - \ell) \right\rangle. \quad (2.91)$$

Since B-splines have compact support, the autocorrelation sequence

$$r_{k-\ell} = \int_{-\infty}^{\infty} N_m(t - k) N_m(t - \ell) dt \quad (2.92)$$

is a finite sequence where

$$r_k = N_{2m}(m - k); \quad k = -m + 1, \dots, m + 1,$$

so that

$$\widetilde{c}_\ell = \sum_k c_k r_{k-\ell} \quad (2.93)$$

can be computed very efficiently. If the final objective of the processing requires the reconstruction of the signal, as in image compression, one needs to eventually convert the processed coefficients back to B-spline coefficients for image quality evaluation and display. However, if the processing objective is for detection and recognition, the dual-spline coefficients can be used for neural network training purposes.

2.7 Decomposition of a 1-D Signal

The decomposition algorithm for scaling function coefficients is described by

$$c_{j,k} = \sum_{\ell} a_{2k-\ell} c_{j+1,\ell}.$$

We can rewrite the expression as

$$c_{j,k} = \sum_{\ell} (a_{m-\ell})|_{m=2k} c_{j+1,\ell}, \quad (2.94)$$

which is interpreted as convolution before downsampling by 2. In terms of digital signal processing symbolism, it is given in Fig 1.8.

The computation of the wavelet coefficients at one level of resolution is carried out in a similar manner, namely,

$$d_{j,k} = \sum_{\ell} (b_{m-\ell})|_{m=2k} c_{j+1,\ell}. \quad (2.95)$$

Combine these two steps to form the decomposition block as shown later. This decomposition block can be repeated and sequentially applied to the scaling function coefficients $c_{j-p,k}$, to yield the wavelet coefficient sequences $d_{j-p-1,k}$, $p = 0, 1, 2, \dots, M$.

Implementation of (2.94) is simple. Taking a scaling coefficient set to convolve with the coefficient set $\{a_k\}$ and downsampling yields the scaling function coefficients at one lower resolution level. Repeating the same procedure with coefficient set $\{b_k\}$ yields the wavelet coefficients. These procedures are repeated to yield the coefficients at lower resolution levels.

2.8 Reconstruction of 1-D Signals

The two-scale relations for the approximation space and the wavelet space constitute the reconstruction algorithm. The scaling function coefficients at a higher resolution level are computed by using the formula

$$c_{j+1,\ell} = \sum_k [p_{\ell-2k} c_{j,k} + q_{\ell-2k} d_{j,k}].$$

Each summation can be interpreted as a convolution process after up-sampling. This process is depicted in Fig 1.8. It can be repeated for coefficient sequences $\{c_{j-p,k}\}$ and $\{d_{j-p,k}\}$, $p = M, M-1, \dots, 0$. The reader should use the Toolware to practice this algorithm for 1-D signals.

2.9 Thresholding

Thresholding is one of the most often used processing tools in wavelet signal processing. It is used in noise reduction, in signal and image compression, and sometimes in signal recognition. The four types of thresholding we use are (1) hard thresholding, (2) soft thresholding, (3) quantile thresholding, and (4) universal thresholding. The choice of thresholding methods depends on the application. We discuss each type here briefly.

2.9.1 Hard thresholding

Hard thresholding sometimes is called gating. If a signal (or a coefficient) value is below a preset value, it is set to zero. That is,

$$\begin{aligned} y &= x, \text{ for } x \geq \sigma \\ &= 0, \text{ for } x < \sigma, \end{aligned} \quad (2.96)$$

where σ is the threshold value or the gate value.

2.9.2 Soft thresholding

Soft thresholding is defined as

$$\begin{aligned} y &= f(x - \sigma), \text{ for } x \geq \sigma \\ &= 0, \text{ for } x < \sigma. \end{aligned} \quad (2.97)$$

The function $f(x)$ generally is a linear function (a straight line with slope to be chosen). However, spline curves of third or fourth orders may be used to effectively weight the value greater than σ .

In some applications such as signal compression, using a quadratic spline curve may increase the compression ratio by a certain amount.

2.9.3 Quantile thresholding

In certain applications, such as image compression, where a bit quota has been assigned to the compressed file, it is more advantageous to set a certain percentage of wavelet coefficients to zero to satisfy the quota requirement. In this case, the setting of the threshold value σ is based on the histogram and total number of coefficients. The thresholding rule is the same as hard thresholding.

2.9.4 Universal thresholding

In some noise removal applications in which the noise statistics is known, it may be more effective to set the threshold value based on the noise statistics. For example, Donoho and Johnstone [8] set the threshold value to be

$$\sigma = v \sqrt{\frac{2 \log(\ell)}{\ell}}, \quad (2.98)$$

where v is the standard deviation of the noise and ℓ is the cardinality of the data set. This threshold value can be used in either hard or soft thresholding as shown earlier.

Implementation

Implementations of the hard, quantile and universal thresholding methods are quite simple. One simply subtracts the threshold value from the magnitude of each coefficient. If the difference is negative, the coefficient is set to zero. If the difference is positive, no change is applied to the coefficient. To implement the soft thresholding by using a linear function of unit slope, the thresholding rule is

$$\begin{aligned} y &= x - \sigma \text{ if } x - \sigma \geq 0, \\ &= 0 \text{ if } x - \sigma < 0. \end{aligned} \tag{2.99}$$

2.10 Two-Dimensional Extension of Wavelet Algorithms

We have mentioned in previous sections that the 2-D wavelets are tensor products of the 1-D scaling function and the wavelet. Corresponding to the scaling function ϕ and the wavelet ψ in one dimension are three 2-D wavelets and one 2-D scaling function at each level of resolution. As a result, the 2-D extension of the wavelet algorithms is the 1-D algorithm applied to both the x and y directions of the 2-D signal. Let us consider a 2-D signal as a rectangular matrix of signal values. In the case where the 2-D signal is an image, we call these signal values Pixel values corresponding to the intensity of the optical reflection. Consider the input signal $c^j(m, n)$ as an $N \times N$ square matrix. We may process the signal along the x direction first. That is, we decompose the signal row-wise for every row using the 1-D decomposition algorithm. Because of the downsampling operation, the two resultant matrices are rectangular, and of size $N \times \frac{N}{2}$. These matrices are then transposed and processed row-wise again to obtain four $\frac{N}{2} \times \frac{N}{2}$ square matrices, namely, $c^{j-1}(m, n)$, $d_1^{j-1}(m, n)$, $d_2^{j-1}(m, n)$, and $d_3^{j-1}(m, n)$. The subscripts of the d matrices correspond to the three different wavelets. This procedure can be repeated an arbitrary number of times to the $c^\ell(m, n)$ matrix (or the LL component), and the total number of coefficients after the decomposition is always equal to the initial input coefficient N^2 . The reader may use the Toolware to practice the 2D decomposition and reconstruction algorithms.

If the coefficients are not processed, the original data can be recovered exactly through the reconstruction algorithm. The procedure is simply the reverse of the decomposition, except that the sequences are

$\{p_k, q_k\}$ instead of $\{a_k, b_k\}$. Care should be taken to remember upsampling first before convolution with the processing sequences.

2.11 Other Algorithms

Toolware also includes following codes for wavelet processing:

- Noise removal using hard thresholding
- Image compression using thresholding
- 1-D wavelet packet decomposition and reconstruction
- 2-D wavelet packet decomposition and reconstruction

Implementation of noise removal is straightforward and is not elaborated here. The user can make use of the Toolware to see the effect of speckle noise being removed from an image, using hard thresholding.

Setting the wavelet (or wavelet packet) coefficients at small amplitudes to zero by thresholding compresses the image information because it takes many fewer bits to represent clusters of zero coefficients. The reconstructed image will lose fidelity because some high frequency components of the image have been removed by thresholding.

The algorithmic logic of the 1-D and 2-D wavelet packet algorithms has been discussed in Part I. The basic algorithm is no different from the wavelet algorithms, except the user has to keep track of the decomposition structure. A procedural discussion of these algorithms is very lengthy and will not be presented here. We encourage the user to practice this algorithm in the Toolware using the images provided, or images from other sources.

Part III

Toolware User's Manual

This page intentionally left blank

3.1 The Wavelet Toolware Overview

Wavelet Toolware is a 32-bit, stand-alone software program with all necessary library routines bundled together in a package. It has five tools: the one- and two-dimensional (1-D/2-D) discrete wavelet transform (DWT) tools, the continuous wavelet transform (CWT) tool, the short time Fourier transform (STFT) tool, and the wavelet generation tool. 1-D/2-D DWT tools respectively support DWT decomposition, DWT coefficient processing, and DWT reconstruction of one- and two-dimensional signals. The CWT tool reads in a 1-D signal data file, computes its CWT coefficients, and then displays the coefficients on the screen. Similarly, the STFT tool reads in a 1-D signal and generates the STFT coefficients for display. The wavelet generation tool is a simple utility to generate B-spline wavelets and Daubechies orthonormal wavelets.

The 1-D and 2-D DWT tools have a similar operating structure. They assist the user to decompose a signal, using scaled wavelets at different levels, process DWT coefficients, and then reconstruct the signal in a very flexible manner. A decomposed signal can be reconstructed from any level to any other lower level, with only selected levels affected. In addition to some simple processing functions already included in Toolware, the user can also add new functions to Toolware through a defined dll program interface. Toolware creates new files only at the explicit request of the user. Thus, accidental crash or termination of Toolware execution should not affect existing files. Toolware runs on the Microsoft Win95 or NT (3.1, 3.5, 3.51, 4) operating system. For performance and stability reasons, we recommend systems with a Pentium 133Mhz or equivalent processor, and 32MB of main memory. The memory size should proportionally increase for processing of large files to keep intermediate results.

3.1.1 Installation of Toolware

Toolware installation is very simple. The user just follows these steps, and the installation program does the rest.

1. Insert the CD-ROM into the CD drive E, which is the most typical CD drive label. If the label of the CD drive on your machine is not E, use the proper CD drive label in subsequent steps.
2. Click on the *run* icon on the desktop.
3. Type in *E:setup* command, and the installation will start.

- 4. Interact with the installation software to load Toolware program files and data files into the directory that you want to host Toolware.

3.1.2 Wavelets

Toolware supports several different wavelets, including Daubechies orthonormal wavelets (Daub1 to Daub10); linear B-spline wavelet and cubic B-spline wavelet; Battle–Lemarie spline wavelets; zeroth- and first-order Meyer wavelets; and biorthogonal Daubechies wavelets (1,1) to (1,3), (2,1) to (2,4), (3,1) to (3,5), (4,1), (4,2), and (5,1). As shown in Fig 3.1, an image is decomposed using the tensor product of the 1-D transform of an image along its vertical and horizontal directions, respectively. Boundary data for both 1-D and 2-D data are treated by a simple wrap-around or data-reflection method to achieve perfect (or nearly perfect) reconstruction.

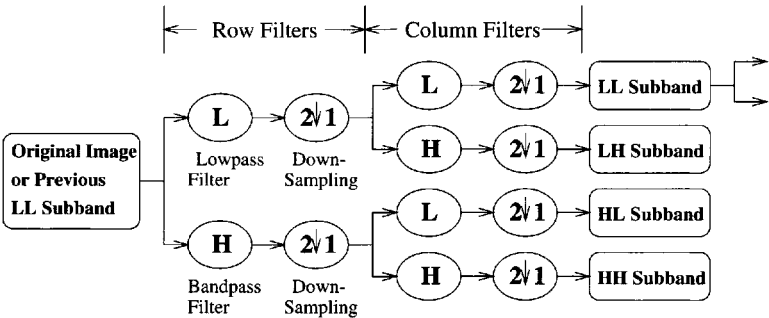


Figure 3.1: The one-level DWT decomposition structure of a 2-D image in Toolware

Toolware is not designed for solving high-precision computational problems, but is meant for general learning and experimentation. Typical numerical errors in the decomposition and reconstruction of a 512×512 gray-scale image are listed in the following table. The signal-error ratio is defined as the absolute ratio of the reconstruction errors divided by a signal, expressed in decibels. Our test results show that the difference in numerical errors on different machines is very small.

wavelet	errors	signal-error ratio
Daub1	0.00000e+00	∞ dB
Daub2	8.16986e-20	2.39009e+02 dB
Daub3	1.63352e-18	2.26000e+02 dB

wavelet	errors	signal-error ratio
Daub4	7.98535e-20	2.39108e+02 dB
Daub5	1.15028e-19	2.37523e+02 dB
Daub6	1.17830e-19	2.37418e+02 dB
Daub7	1.17253e-19	2.37440e+02 dB
Daub8	4.24760e-19	2.31849e+02 dB
Daub9	7.74388e-20	2.39241e+02 dB
Daub10	8.43506e-19	2.28870e+02 dB
BiDaub1_1	0.00000e+00	∞ dB
BiDaub1_2	0.00000e+00	∞ dB
BiDaub1_3	0.00000e+00	∞ dB
BiDaub2_1	0.00000e+00	∞ dB
BiDaub2_2	0.00000e+00	∞ dB
BiDaub2_3	0.00000e+00	∞ dB
BiDaub2_4	8.46053e-13	1.68857e+02 dB
BiDaub3_1	0.00000e+00	∞ dB
BiDaub3_2	0.00000e+00	∞ dB
BiDaub3_3	0.00000e+00	∞ dB
BiDaub3_4	5.28276e-01	5.09022e+01 dB
BiDaub3_5	8.46053e-13	1.68857e+02 dB
BiDaub4_1	4.90989e-20	2.41220e+02 dB
BiDaub4_2	1.14181e-21	2.57555e+02 dB
BiDaub5_1	4.92118e-20	2.41210e+02 dB
Linear BSpline	7.94940e-08	1.19127e+02 dB
Cubic BSpline	3.08383e-01	5.32399e+01 dB
Battle_Lemarie	1.10625e-17	2.17692e+02 dB
Meyerzero	6.31288e-02	6.01285e+01 dB
Meyerfirst	4.62319e-02	6.14814e+01 dB

3.2 Using Toolware

Toolware is activated by double-clicking on the Toolware group icon, Fig 3.2, which is placed at the designed directory during installation, to bring out the main menu, Fig 3.3. After the main menu appears on the screen, the user can activate any of the five tools simultaneously. The five tools include one-dimensional discrete wavelet transform and processing (), two-dimensional discrete wavelet transform and processing (), continuous wavelet transform (), short time Fourier transform (STFT) (), and the generation of



Figure 3.2: The start-up icon for Toolware programs

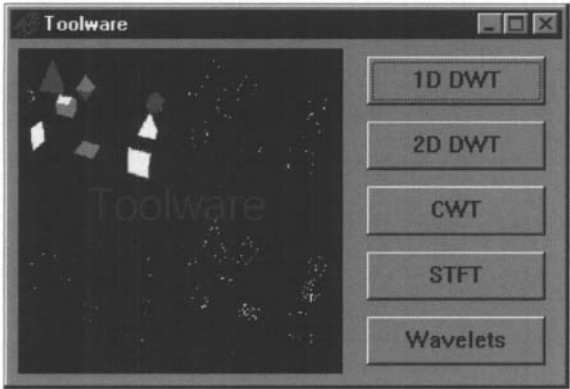


Figure 3.3: Toolware main menu

wavelets (**Wavelets**). These tools operate independently, and they use different data formats.

3.3 2-D DWT Tool

After you open the 2-D DWT tool, you need to choose a data file before proceeding with other processing functions. This is done by clicking on the (**File**) button.

3.3.1 Data file selection

Toolware accepts pgm image files, meaning that any files appended with “.pgm” can be read into Toolware for processing. After the user clicks on the File button, or its corresponding graphical icon, the file selection window (Fig 3.4) will pop out for the user to open an image file for processing. From the file window, one can identify any image file (in PGM format) in any directory and then load the file by clicking on the  button.

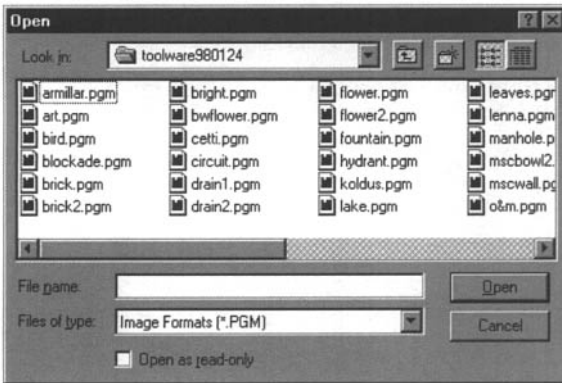
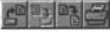


Figure 3.4: File selection window for 2-D images

A loaded image is automatically displayed on the screen for DWT and processing (see Fig 3.5 for a sculpture image, file name “art.pgm,” taken from the main campus of Texas A&M University). The left-most corner of this window lists the title of the tool and the chosen file name. The two buttons at the upper right corner ( ) maximize, minimize, and close the current window. The second row (   ) lists the available functions in the 2-D DWT tool where each of these headings is a pull-down menu.

Although they could be activated from the pull-down menu, key operations of the 2-D tool are represented by thumbnail icons at the third row of its window.



The first group of four icons  enable opening, saving, and printing a displayed image. The second group of three icons  start the decomposition, reconstruction, and processing of the selected image. Up to five levels of DWT coefficients can be stored in the internal buffer for display, but only levels 0 to 3 can be processed by processing functions through the P button. By clicking on one of the following icons (    ), the user can display any particular decomposi-

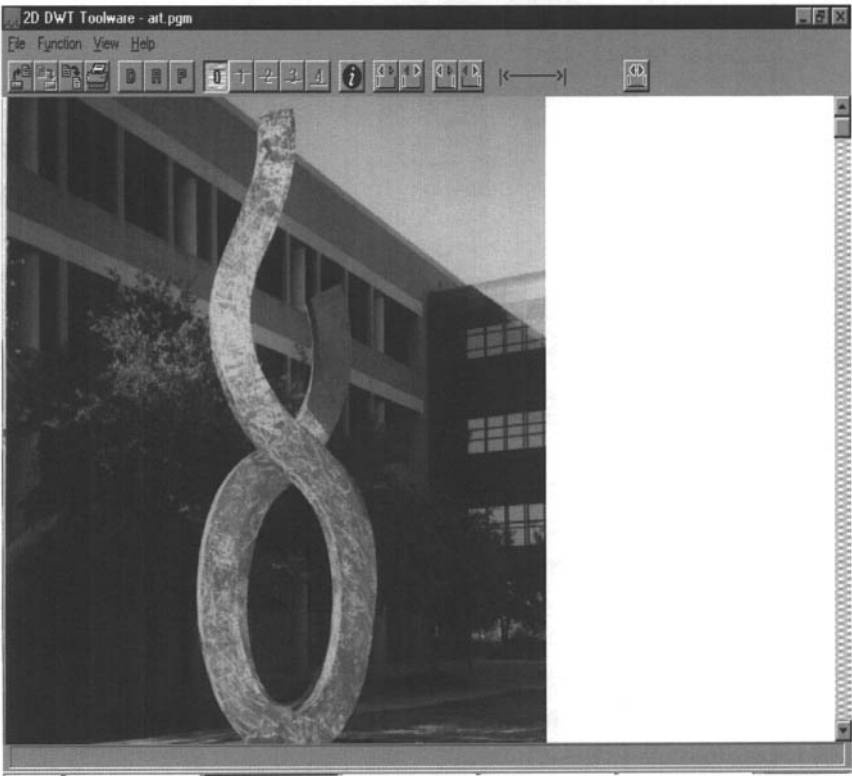


Figure 3.5: A 2-D image in the 2-D tool

tion level of wavelet coefficients.

All DWT coefficients at different levels are buffered in Toolware, and they are dynamically packed together to form one display image as needed. Level 0 represents the original data, or the image reconstructed from its DWT coefficients. Obviously, the reconstructed image can be different from the original one if certain DWT coefficients are manipulated by processing functions in the tool. The user needs to take precaution in preserving the original image while saving new results into the hard disk.

3.3.2 Wavelet decomposition

After an image is loaded into the 2-D tool, it can be decomposed by clicking on the  button. Then, a list of wavelet decomposition sequences will pop up along with a choice of the decomposition structure. Fig 3.6 shows an example, in which a five-level wavelet-packet structure is selected for decomposition of an image using the Daubechies biorthogonal wavelet. The decomposition process is started and terminated automatically by clicking on the OK button.

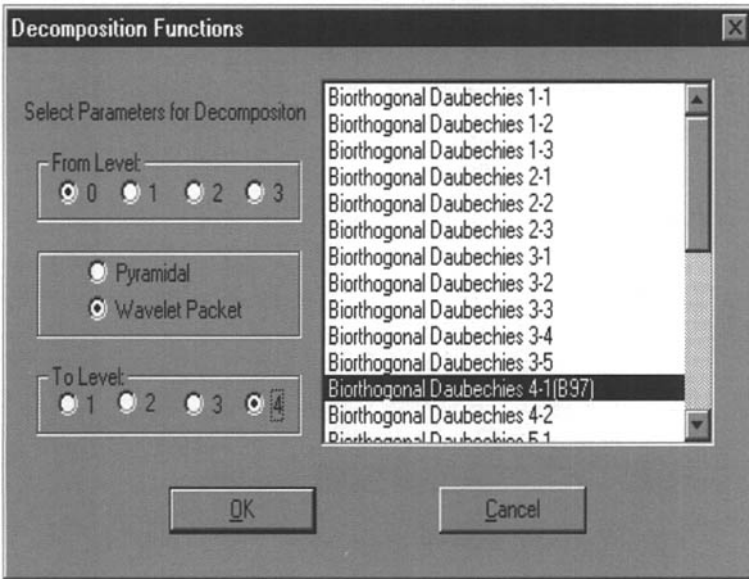


Figure 3.6: Selection of the decomposition parameters

After Toolware has performed the decomposition based on selected parameters, the decomposed image will pop up on the window, so that the user can view the absolute value of the wavelet coefficients, see Fig 3.7. The displayed image is generated by using the truncated absolute

values of wavelet coefficients, but raw DWT coefficients are retained in an internal buffer for actual processing. As needed, any other decomposition level within the decomposition range can be seen by clicking on one of the icons (). When the level i is clicked on, all DWT coefficients from level 1 to level i are displayed in the same window. In a composed image of DWT coefficients, the upper left block is the LL (low–low) band, the lower left block is the LH (low–high) band, the upper right block is the HL (high–low) band, and the lower right block is the HH (high–high) band. Although the image is decomposed for four levels, the example in Fig 3.7 shows that we can still retrieve level-2 wavelet packet coefficients for processing and display, since all coefficients at every level are saved into an internal buffer. The screen remains unchanged if an invalid level of an image is chosen. This can happen, for example, if the user tries to view the second-level DWT coefficients of an image not yet decomposed.

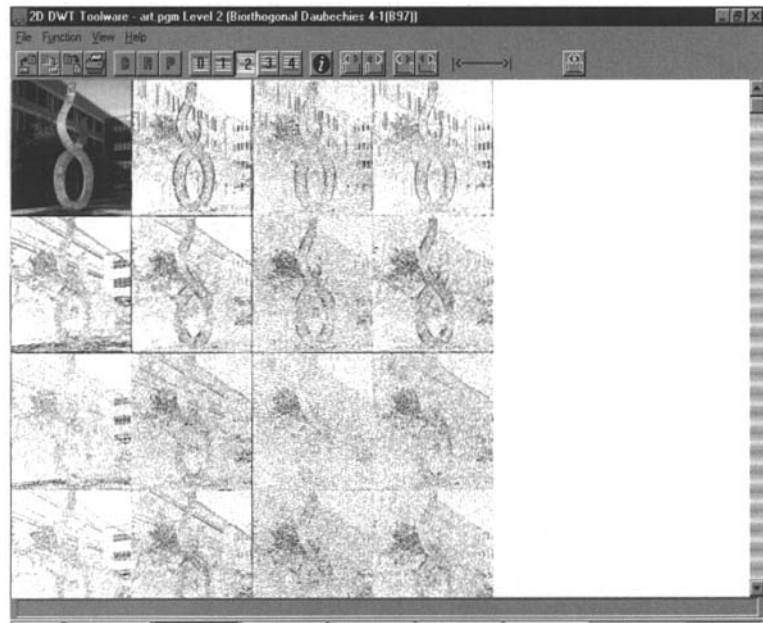
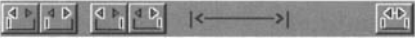


Figure 3.7: DWT coefficient display

3.3.3 Display thresholding

Wavelet coefficients are usually very sparse. For much better visual effects, we display wavelet coefficients, except for the coefficients in the lowest subband, in reverse brightness. That is, except for DWT coefficients in the lowest subband, i.e., LL, LLLL, LLLLLL, etc., larger

DWT coefficients are represented by darker display pixels. For convenience of operation, a set of buttons, (), is available for quick preview of thresholding effects without changing their internal values. The two leftmost buttons decrease/increase the smallest (absolute) values to be displayed. The next two buttons decrease/increase the largest (absolute) values to be displayed. Next to them, the ruler-like icon indicates the selected lower/upper bounds with respect to the range of true values. Wavelet coefficients that fall into the lower/upper bound setting are scaled to the display range of 0 to 255. The far right button resets all the threshold settings and returns the image display back to its original form. We remark here that, unlike the processing functions, the display thresholding does not have permanent effects on the internal values of the displayed DWT coefficients. The user should use built-in or custom processing functions to alter DWT coefficients for actual signal processing purposes. Fig 3.8 shows an example in which some DWT coefficients are thresholded in the display buffer, so that they become less visible on the screen.

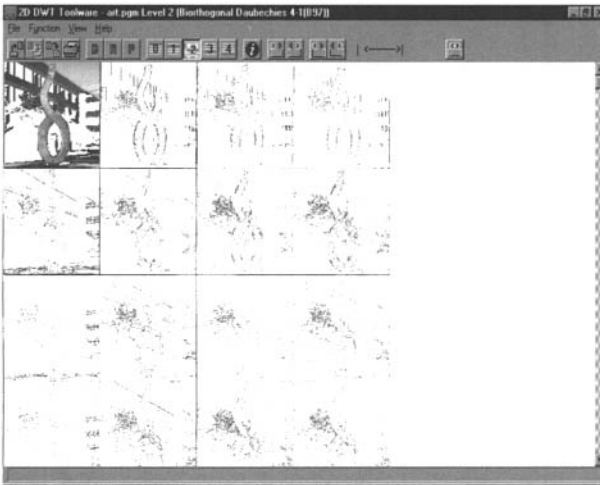


Figure 3.8: DWT coefficient display after thresholding

3.3.4 Wavelet reconstruction

We note that once a data set is decomposed, its selected decomposition parameters are locked into the tool to prevent erroneous, meaningless reconstruction. When clicking on the  button, a window for selection of reconstruction parameters will pop up. Toolware can perform reconstruction between different levels, and their results will be stored

in its internal buffers with the possibility of overwriting their existing contents. For example, if an image is decomposed into four levels, its DWT coefficients at levels 1, 2, 3, and 4 are kept in the internal buffers. If a level-3 to level-0 reconstruction request is selected, as in Fig 3.9, DWT coefficients stored in level 3 buffers are retrieved and then reconstructed back to level-2 subbands. The new level-2 values will be saved into the level-2 buffers. Then, level-2 coefficients will be reconstructed into level-1 coefficients, and new level-1 coefficients will be reconstructed back to level-0. If any of level-3 coefficients have been altered before the reconstruction request, it is likely that old level-2, level-1, and level-0 coefficients will be overwritten in the reconstruction process.

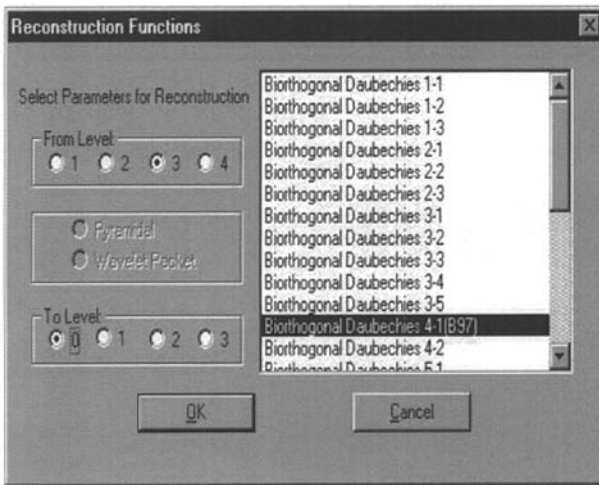


Figure 3.9: Selection of DWT reconstruction

As usual, the user can view the new DWT values after reconstruction. If the user exits Toolware at this point, the original file on the hard disk is not affected, but if the file-save button is invoked, then the latest level-0 image will replace the original image if the same file name is chosen.

Coefficient processing

In most signal processing applications, DWT coefficients are altered, deleted, or clustered to achieve certain effects. The 2-D tool allows the user to add new functions for processing of DWT coefficients in its buffers. Processing of DWT coefficients is triggered by the  button, and a list of processing functions will pop up.

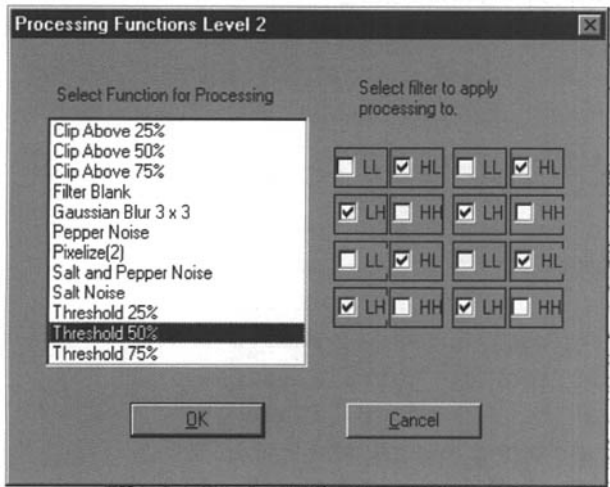


Figure 3.10: Selection of DWT processing functions

In the example shown in Fig 3.10, the user has chosen to apply a simple thresholding routine to eight subbands of the decomposed art.pgm image. When the OK button is clicked on, this routine will be applied to each of the subbands, one after another, until all the selected subbands are processed. Post-processed DWT coefficients will be displayed on the screen automatically, see Fig 3.11.

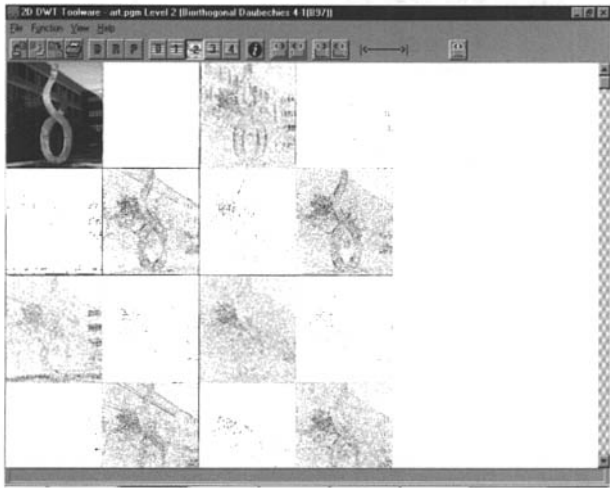


Figure 3.11: Post-processed DWT coefficients

In addition to its own internal routines, the 2-D tool will accept user-defined processing functions, provided that they conform to the following “dll” interface structure. More detail on the integration of

processing functions to the Toolware will also be discussed in the section on the 1-D tool.

```
#define PROC2D /*Tell the Toolware that this
                is a 2-D processing routine*/
#define NAME "Simple Smoothing" /*The name of
                                this routine*/
#include "extfunc1.h" /*Some Toolware symbols*/
/* This routine accepts floating point numbers
   placed in the 2-D array "input",*/
/* process the array, and then place the results
   at the 2-D array "output". */
/* The dimension of the matrix is specified by
   "x" and "y".*/
TOOLBOX_FUNCTION process(float huge **input,
float huge **output, int x, int y)
{
    int i,j;
    for(i=1;i<x-1;i++)
        for(j=1;j<y-1;j++)
            output[i][j]=
                (input[i-1][j-1] +input[i][j-1]
                 +input[i+1][j-1])+
                (input[i-1][j] +input[i][j]
                 +input[i+1][j])+
                (input[i-1][j+1] +input[i][j+1]
                 +input[i+1][j+1])/9.0;
}
```

One needs to use a compiler to compile a new routine into a Windows "dll" (dynamic link library) file and place it into the Toolware directory. Otherwise, the 1-D tool will not be able to find your custom routine for execution. In using Borland C++ 5.0, one loads the .ide project (in C++) into the compiler and makes necessary functional changes. Then, simply build the dll library to be saved into the Toolware directory. If you use other compilers to generate codes, in order to be compatible, the .dll used must have names exported explicitly (see dll.map). If these files do not match, Toolware will assume that the .dll is not an external tool designed for it and will ignore it.



Figure 3.12: 1-D data file selection

3.4 1-D DWT Tool

After the 1-D DWT tool is activated, it does not display any signal before any data file is opened. The 1-D DWT tool has the same set of control buttons as the 2-D DWT tool, but their display and processing mechanisms are quite different. The list of files that can be read into the 1-D DWT tool by clicking on the **File** button is displayed on the screen, see Fig 3.12.

Data file format

The 1-D DWT tool accepts numerical (integer or floating-point) files in the ASCII format. Appending “pts” to a file name, i.e., *file_name.pts*, makes it visible to the 1-D DWT tool, and the file name will appear on the file list of Toolware. The following are three examples that have an acceptable data format for the Toolware.

Style 1:

```
-----
0.000000
0.000111
0.000444
0.000998
0.001773
0.002765
0.003973
0.005395
0.007027
0.008866
```

0.010906
0.013145
0.015577

Style 2:

0.000000 0.000111 0.000444 0.000998 0.001773
0.002765 0.003973 0.005395 0.007027 0.008866
0.010906 0.013145 0.015577

Style 3:

37 40 204 80 88 163 186 131 112 157 129 120 82 69
116 74 43 73 114 73 90 123 147 149 165 193 185 168
180 203 172 165 181 182 183 177 191 210 197 219 224
148 96 255 100 1 144 88 20 7 55 45 27 74 43 58 193
74 93 179 185 113 114 172 114 10 0 87 70 100 64 38
75 109 63 85 139 135 138 161 172 188 186 173 185 206
170 163 1 93 182 171 178 196 208 194 214 223 191 74
226 181 0 123 109 33 5 42 56 26 49

After a data file is loaded into the Toolware, it is graphically displayed on the screen in a three-row format, see Fig 3.13, and the data is ready for DWT decomposition, processing, and reconstruction. The three rows have three different buffers, denoted as DB_1 , DB_2 , and DB_3 , which will be also used for processing and reconstruction of DWT coefficients. In this example, both DB_1 and DB_2 contain an unprocessed signal, and DB_3 is empty. Details on management of the three display buffers will be explained later.

3.4.1 1-D DWT decomposition

The 1-D tool supports both pyramidal decomposition and wavelet-packet decomposition. Similar to the decomposition procedure of the 2D DWT Tool, you need to first specify a decomposition level and a wavelet type for decomposition of a one-dimensional signal. The 1-D DWT decomposition is activated by clicking on . A list of wavelets will pop up on the screen together with decomposition structures, see Fig 3.14. At the user's choice, one needs to select the wavelet type, the decomposition level, and the decomposition structure, i.e., pyramidal or wavelet packet, and then click on the OK button to start decomposition. In this example, we have chosen a level-0 to level-4,

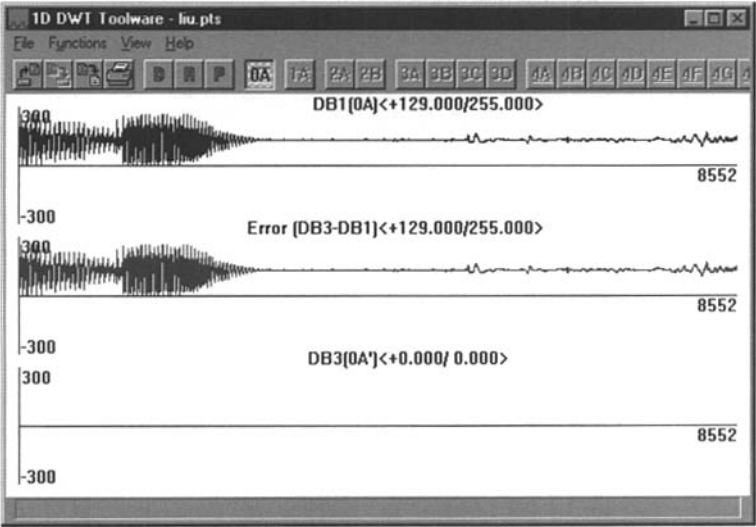


Figure 3.13: The three-row display of an acoustic signal

wavelet-packet decomposition of the signal by the cubic-spline wavelet.

3.4.2 Viewing and processing the data

The 1-D DWT tool can decompose a signal up to four levels. As in the 2-D tool, it also uses different buffers to store DWT coefficients. For example, if a signal is decomposed for three levels, its DWT coefficients in any of the three levels can be retrieved for processing, reconstruction, and display. When a particular subband at a level is chosen by clicking on one of the following “level” buttons, the DWT coefficients will be loaded into the three display buffers DB_1 , DB_2 , and DB_3 for

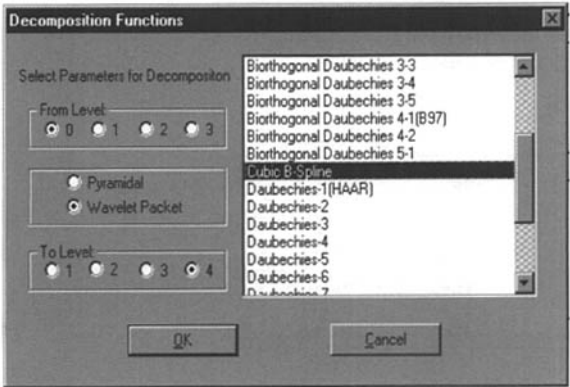


Figure 3.14: Selection of decomposition parameters

visualization based on the following rules.



When the user clicks on 0A, the original signal is displayed on DB_1 and the reconstructed signal, if any, is displayed on DB_3 . The (reconstruction) difference between DB_1 and DB_3 is shown at DB_2 . If the system does not have any reconstructed data yet, then DB_3 (and thus the third display row) is empty, and both DB_1 and DB_2 display the original data because DB_2 keeps the difference between DB_1 and DB_3 . For other decomposition levels, DB_3 displays the parent subband of the two chosen subbands displayed in DB_1 and DB_2 . That is, at level 1A, which represents the two subbands decomposed from the original signal at 0A, its low-pass and bandpass subbands are displayed in DB_1 and DB_2 , respectively. The parent of 1A, which corresponds to the original signal, is displayed on DB_3 . For following levels, letters A and B respectively denote the low-pass and bandpass children of a set of DWT-coefficients.

When level 2A is chosen, the low-pass and bandpass children of the level 1A low-pass subband, which is displayed on DB_3 , are respectively displayed on DB_1 and DB_2 . Similarly, at level 2B, the low-pass and bandpass children of the level 1A bandpass subband, which is displayed on DB_3 , are respectively displayed in DB_1 and DB_2 . Rules for use of DB_1 and DB_2 at other levels are listed as follows.

Level	DB_1 and DB_2 display
3A	children from the low-pass band of 2A
3B	children from the bandpass band of 2A
3C	children from the low-pass band of 2B
3D	children from the bandpass band of 2B
4A	children from the low-pass band of 3A
4B	children from the bandpass band of 3A
4C	children from the low-pass band of 3B
4D	children from the bandpass band of 3B
4E	children from the low-pass band of 3C
4F	children from the bandpass band of 3C
4G	children from the low-pass band of 3D
4H	children from the bandpass band of 3D

Fig 3.15 shows DWT coefficients of an signal. In its normal display mode, the maximum and minimum values of all DWT coefficients, in-

cluding the original data, are used to set the display scales. This usually leads to loss of visual details of small coefficients in some subbands, as expected, for the sake of display consistency. To remedy this problem, the 1-D tool is equipped with a sliding *magnifier* through which one can better view details of small coefficient values. By simply pressing on the left mouse button, a magnified version of data surrounding the cursor will display on a small floating window, see Fig 3.16. The zoom-in window disappears at the release of the button.

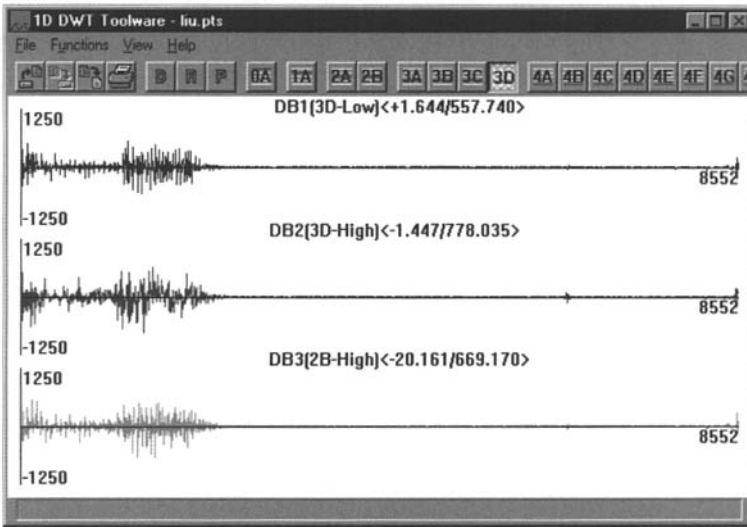


Figure 3.15: The three-row display of DWT coefficients at level-3

For a given display level, the user can process data stored in the DB_1 to DB_3 by clicking on the “P” (processing) button, , and then the list of processing functions will pop up, see Fig 3.17. By clicking on a processing function together with one or more buffer buttons (uppermost signal (DB_1), middle signal (DB_2), bottom signal (DB_3)), the user can process data stored in selected buffers by the chosen function. Both the internal and display buffers are refreshed after the processing function is executed.

At any time, the user can reconstruct all or some DWT coefficients. Similar to the 2D Tool, once a signal is decomposed, its decomposition structure is registered into Toolware, and the user can only use the selected decomposition wavelet and structure for reconstruction. In the example shown in Fig 3.18, only the reconstruction levels can be chosen by the user, and all other parameters will be set by Toolware. Once the reconstruction command is executed, new level-3 DWT coefficients are reconstructed from level-4 DWT coefficients and saved into level-3

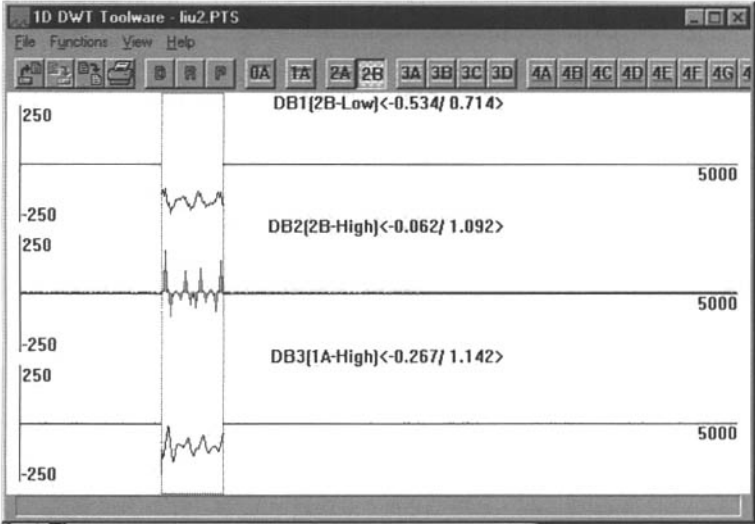


Figure 3.16: A magnified view of small DWT coefficients

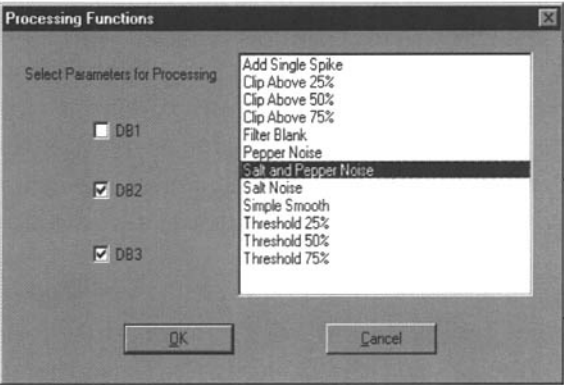


Figure 3.17: The window for 1-D DWT processing functions

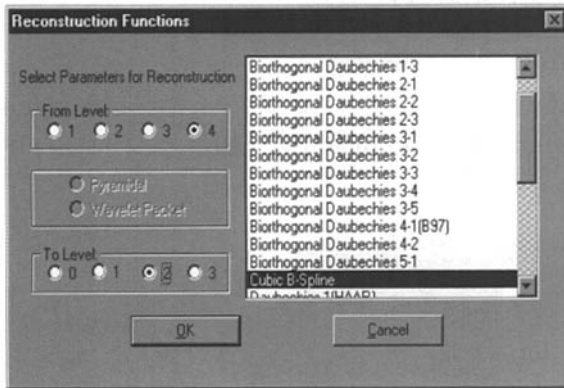


Figure 3.18: Selection of the reconstruction structure

buffers. Level-2 buffers are updated in a similar fashion. This process repeats until all levels between the chosen reconstruction levels are updated. As usual, you can view the modified signal after reconstruction.

Custom processing routines

Like the 2-D tool, the 1-D tool also accepts custom routines for processing wavelet coefficients. The following is an example routine for smoothing of the chosen signal.

```
#define PROC1D    /*Tell the Toolware this
                                     is a 1-D routine*/
#define NAME "Simple Smoothing"    /*Name of the routine*/
#include "extfunc1.h"    /*Toolware symbols*/
/* This routine accepts floating point numbers placed in
the array "input",*/
/* process the array, and then place the results at the
array "output". */
/* The total number of data samples is defined by "len".*/
TOOLBOX_FUNCTION process(float huge *input,
float huge *output, int len)
{
    int i;
    for(i=1;i<len-1;i++)
        output[i]= (input[i-1] +input[i] +input[i+1])/3.0;
}
```

3.5 Continuous Wavelet Transform Tool

The continuous wavelet transform tool (CWT tool) generates the continuous wavelet coefficients of a one-dimensional signal. It has eight different control buttons, see Fig 3.19.



Figure 3.19: CWT tool control buttons

Starting from left to right, the first button is for selection of a data file for processing, the second and third are for saving the displayed CWT coefficients into an ASCII data file, or a pgm (image) file, respectively. The image of the CWT coefficient matrix can be printed to your printer by clicking on the printer button. The relative display intensity of CWT coefficients can be adjusted by the next four buttons. Clicking on the fifth button, which has a straight line, means that the display intensity directly reflects the magnitude of CWT coefficients. The next two buttons are for (de)emphasis of small and large coefficients. The eighth button allows you to display either only the magnitude, which is the default mode, or signed amplitudes, with different colors assigned to positive and negative values.

The CWT coefficients of the signal, which are organized into a two-dimensional matrix, are graphically shown on the display window of the CWT tool. The x -axis of the display is the time line, and the y -axis is the scale of the signal. After you invoke the CWT tool from the main menu, it will display an empty window. By clicking on the file selection button, you can select a “.pts” or “.wav” file as the signal input. Then, the CWT tool will produce the CWT coefficients through a self-explanatory sequence. After the signal is processed, its CWT coefficients will be displayed on the window.

In this example displayed in Fig 3.20, the data set (file: delta.pts) represents two compounded sinusoidal signals, in addition to two impulses. The frequencies of the two sinusoidal signals are 500 and 1000 Hz, respectively. The magnitudes of the two impulses are located at $t = 0.192$ and 0.196 , respectively, and their magnitude is 6. The sampling rate for this signal is set at 8 kHz. You can see from the “image” that the CWT coefficients of the two sinusoidal signals are clearly displayed horizontally, and those of the two impulses are displayed vertically at their time instants.

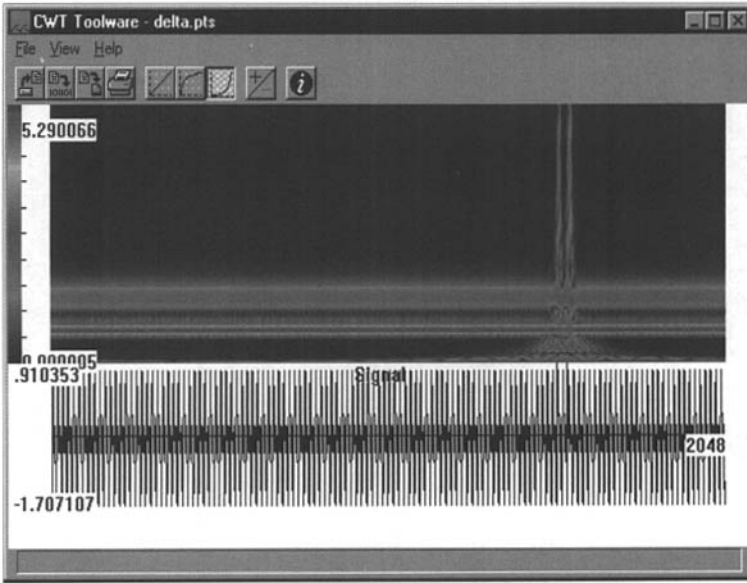


Figure 3.20: CWT coefficients

3.6 Short Time Fourier Transform Tool

Similar to the CWT tool, the STFT tool generates the short time Fourier transform coefficients of a one-dimensional signal, for a given window size. Except for the selection of the window size, which determines the size of a signal segment for FFT computations, the STFT tool has an identical set of control buttons, except for selection of the short time window, and they serve the same purposes as those in the CWT tool. By clicking on the “file” button, you can choose a one-dimensional data file, in either the pts format, or the standard wav format, to be processed by the STFT tool. Similar to the CWT tool, the STFT coefficients of the signal are organized into a two-dimensional matrix and then graphically displayed, where the x -axis of the display is the time line, and the y -axis is the frequency axis of the signal.

It is interesting to compare the results obtained from the CWT to those from the STFT. The following two figures represent the results when the window sizes are set to be $w = 8$ (Fig 3.21), and $w = 64$ (Fig 3.22), respectively. As expected, for a given window size, the STFT has either good time resolution or good frequency resolution, but not both. Using the CWT, we can easily determine that the signal delta.pts has two dominating frequencies, and two impulses, from the CWT coefficients. On the other hand, one must use different window sizes to test this signal in order to reach the same or a similar conclusion.

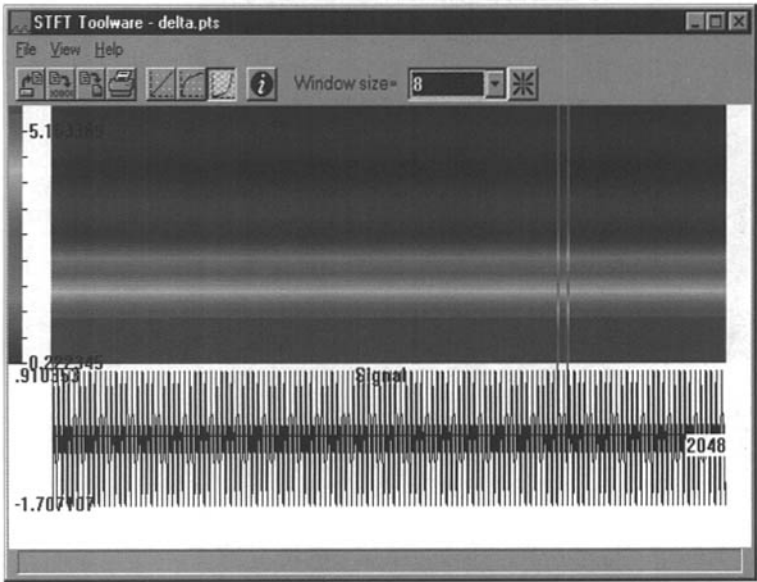


Figure 3.21: STFT coefficients for window size=8

3.7 Wavelet Builder Tool

The wavelet builder tool constructs two different types of wavelets and their scaling functions. In addition to their graphical display, the numerical values of those functions can also be exported to data files. The B-spline functions are generated using their closed-form expressions. By choosing the B-spline family from the file selection button, you will see the Harr wavelet at iteration 1. Each time you click on the yellow, rightward arrow, the wavelet order increases by 1. That means that you will see the linear, cubic, and other splines and wavelets by clicking on the same button. Fig 3.23 displays the cubic spline and its wavelet. The Daubechies orthonormal wavelets are generated in a similar manner. Some precomputed coefficients are already saved in the system for fast retrieval and export. The values of these wavelets functions can be saved into text files for further processing. For example, you could save the numerical values of wavelet functions into data files, and then use the STFT tool to examine their spectra.

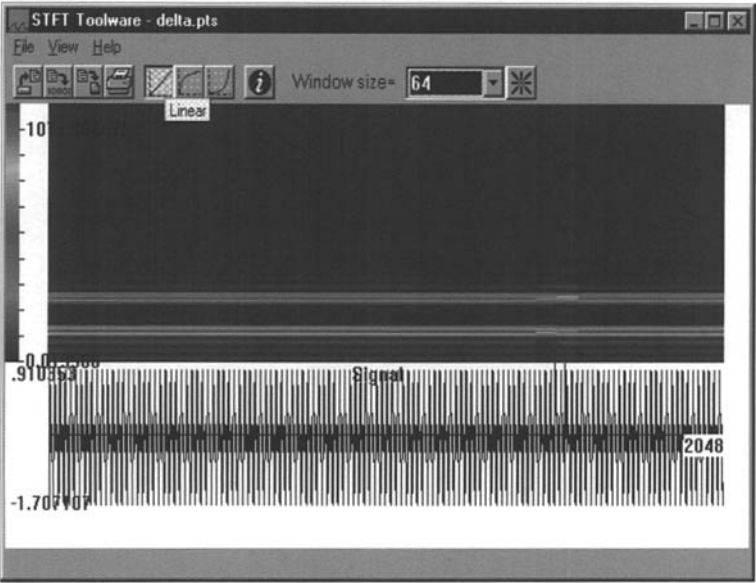


Figure 3.22: STFT coefficients for window size=64

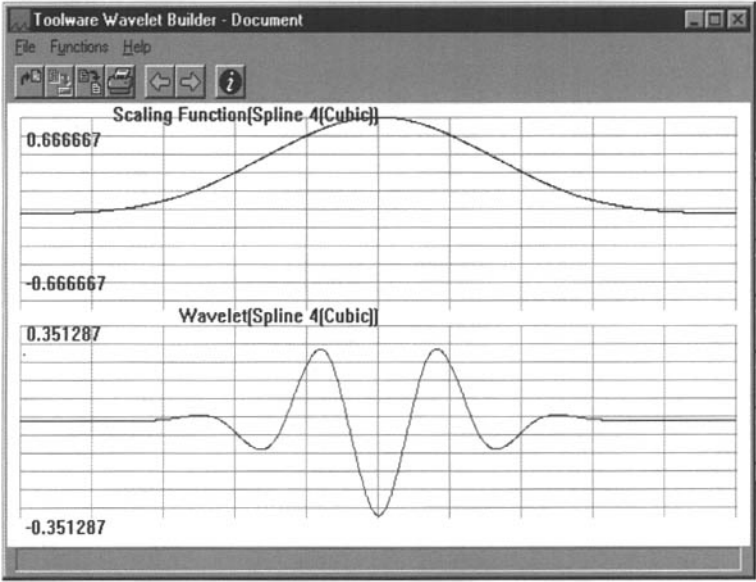


Figure 3.23: Cubic spline and its wavelet

This page intentionally left blank

Bibliography

- [1] Haar, A., Zur Theorie der Orthogonalen Funktionensysteme, *Math Annal.*, **69** (1910), 331-371.
- [2] Chui, C. K., *Wavelet: A Mathematical Tool for Signal Analysis*, SIAM Publ., 1997.
- [3] Chui, C. K. *An Introduction to Wavelets*, Academic Press, 1992.
- [4] Vetterli, M., and Kopvacevic, *Wavelets and Subband Coding*, Prentice Hall Inc., 1995.
- [5] Strang, G., and T. Ngyuen, *Wavelets and Filter Banks*, Wellesley-Cambridge Press, 1996.
- [6] Daubechies, I., *Ten Lectures on Wavelets*, SIAM Publ., (1992), 326.
- [7] Coifman, R. R. and M. V. Vickerhauser, *Entropy-Based Algorithms for Best Bases Selection*, *IEEE Trans. on Information Theory*, **38** (1992), 713-718.
- [8] Donohoe, D. and Johnstone, *Minimax estimation via wavelet shrinkage*, Technical Report, Department of Statistics, Stanford University, (1992).

Index

- N_m , 20
- $\{p_k\}$, 13
- $\{q_k\}$, 13
- $\phi(t)$, 13
- $\psi(t)$, 13
- z -transform, 34
- 2-D signal, 27
- 2-D wavelet decomposition, 27
- 2-D wavelet packet, 27
- 2-D wavelets, 25, 27
- 3db points, 3

- all-pass filters, 15
- analog processing, 18
- antisymmetric, 22
- approximate function, 10
- approximation space, 19, 24, 38
- approximation subspace, 10, 19
- autocorrection sequence, 18
- autocorrelation matrix, 18
- autocorrelation sequence, 37

- B-spline, 20, 35–37
- B-spline functions, 21
- B-spline series, 21, 37
- B-wavelet, 20
- B-wavelet ψ_{N_m} , 20
- bandpass filter, 5
- basic wavelet, 6

- basis function, 9, 13
- basis functions, 6, 15, 19
- Battle–Lamarie wavelets, 20
- Battle–Lemarie, 20
- Best-bases, 24
- best-level, 24
- biorthogonal basis, 19, 22
- biorthogonal wavelet, 22
- biorthogonality, 20
- boundary effect, 31
- broadband signal, 10

- cardinal B-spline function, 20
- chirp signal, 5
- coefficient sequence, 18
- Cohen, 22
- compactly supported, 18, 20
- complex spectrum, 2
- compression, 39
- continuous wavelet transform, 6
- convolution, 2, 16, 38
- correlation, 8
- cubic spline, 13
- CWT, 9

- d-c offset, 5
- Daubechies, 6, 13, 22
- Daubechies orthonormal wavelet, 20

- decomposition, 17, 27
- decomposition algorithm, 17
- decomposition relation, 15, 16
- decomposition sequences, 15, 20, 24
- delta function, 1, 2
- detection, 37
- diagonal bands, 18
- digital processing, 18
- dilation, 11
- discrete Fourier analysis, 1
- discrete Fourier transform, 35
- discrete wavelet transform, 10
- divide and conquer, 10
- downsample, 16, 27, 38
- dual B-spline, 21
- dual B-spline coefficients, 37
- dual B-spline series, 37
- dual scaling function, 20, 21
- dual spline, 21
- dual spline wavelet, 21
- dual wavelet, 20
- dual wavelet spaces, 21
- dual wavelets, 21
- dual-spline, 36, 37
- duals, 36
- energy, 6, 24
- energy concentration, 4
- entropy, 24
- Euler-Frobenius-Laurent Polynomial, 21
- exponential decay, 21
- Fast Fourier Transform (FFT), 30
- Feauvears, 22
- FFT, 31
- filter bank, 10, 18, 22, 35
- finite impulse response, 22
- finite-energy functions, 5
- finitely supported, 22
- FIR filters, 22
- first-order B-spline, 17
- Fourier analysis, 1
- Fourier processing, 6
- Fourier series, 1
- Fourier transform, 1
- frequency bands, 23
- frequency domain, 2, 18
- frequency octaves, 10
- fundamental frequency, 1
- gating, 39
- global periodic function, 1
- Haar, 1
- Haar wavelet, 17
- Haar wavelets, 20
- hard thresholding, 39
- hierarchacal pyramid, 27
- high-high (HH), 27
- high-low (HL), 27
- highpass, 27
- histogram, 39
- infinite product, 34, 36
- infinite sequences, 21
- inner product, 8, 19
- interpolatory, 17
- large scale, 9
- linear B-spline, 17
- linear combination, 6

- linear phase, 22
- linear transform, 6
- LL component, 27
- localization property, 5, 6
- localized finite energy basis, 2
- low frequency, 9
- low-high (LH), 27
- low-low (LL), 27
- lowpass, 27

- main lobe zero-crossing, 3
- Meyer, 20
- Meyer wavelet, 20
- Morlet wavelet, 33
- mother wavelet, 6
- MRA, 10, 15
- multiband signal decomposition, 10

- neural network, 37
- noise reduction, 39
- noise removal, 30, 40, 41
- noise statistics, 40
- normalized, 19
- normalized orthogonal set, 19

- octave bands, 23
- one-variable function, 2
- orthogonal, 19
- orthogonal basis, 21
- orthogonal complementary subspace, 11
- orthogonal projection, 17, 18
- orthogonal subspace, 11
- orthogonal sum, 11
- orthogonality, 19, 20, 22, 37
- orthonormal, 19
- orthonormal basis, 1, 18, 20
- orthonormal set, 19
- orthonormal wavelet basis, 20
- orthonormal wavelets, 6

- Paseval identity, 3
- perfect reconstruction, 20
- periodic signals, 1
- processing filter, 27
- processing sequences, 22
- psuedo-linear phase, 22

- quadratic spline, 39
- quantile thresholding, 39

- recognition, 37
- reconstruction, 17
- reconstruction algorithm, 16
- reconstruction sequences, 20
- reconstructive relations, 15
- rectangular function, 2
- rectangular window, 2
- refinement, 11
- regularity, 35
- Reisz basis, 6
- resolution, 3
- RMS, 3

- sample (discretized) values, 17
- scale, 10
- scaling, 6
- scaling function, 6, 10, 16–18, 20, 34, 35
- scaling function component, 24
- scaling function representation, 17
- scaling function series, 18, 24

- scaling functions, 25
- scaling parameter, 10
- semi-orthogonal, 19, 20
- Short-Time Fourier Transform, 2
- signal compression, 30
- signal distortion, 22
- signal recognition, 39
- similarities, 9
- sinusoidal function, 1, 9
- small wave, 5
- soft thresholding, 39
- speckle noise, 41
- spectral domain, 3–5, 9, 27
- spectral energy, 3
- spectral resolution, 23
- spectral-domain windowing, 3
- spectrum, 6
- spline space, 21
- splitting trick, 23
- STFT, 9, 31
- subspace relation, 15
- symmetric, 22
- tensor product, 40
- tensor-product, 25
- thresholding, 30
- time domain, 2
- time–frequency information, 2
- time-dependent spectrum, 5
- time-domain, 9
- time-domain windowing, 3
- time-frequency, 6
- Toolware, 5
- transient signals, 1, 5
- translate, 6, 20
- translate scaling function, 36
- truncation, 3
- Two-dimensional wavelet packets, 27
- two-dimensional wavelets, 25
- two-scale relation, 13, 20
- two-scale relations, 15, 16, 24
- two-scale sequences, 15, 22
- two-variable function, 2
- uncertainty principle, 3
- unique transformation, 3
- universal thresholding, 39
- up-sampling, 17
- vector space, 19
- wavelet, 5, 6, 11, 16
- wavelet algorithm, 24
- wavelet basis, 6
- wavelet coefficient, 8
- wavelet component, 24, 27
- wavelet decomposition, 23
- wavelet packet, 24, 27
- wavelet packet algorithm, 24
- wavelet packet series, 25
- wavelet packet subspace, 24
- wavelet packet transforms, 23
- wavelet packets, 23, 24
- wavelet series, 18, 24
- wavelet signal processing, 6, 18
- wavelet space, 21, 38
- wavelet subspace, 10, 11, 19
- wavelets, 35
- window area, 9
- window function, 2, 3, 9, 30
- window measures, 9

- window width, 9
- window widths, 9, 10
- Windowed Fourier Transform, 2
- windowing effect, 3

- z-transform, 21